

AD A052614

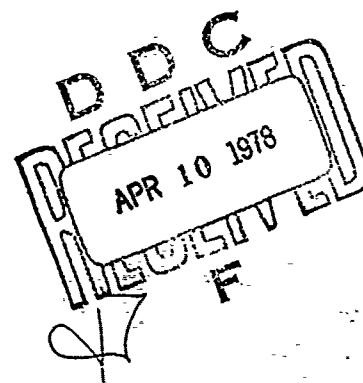
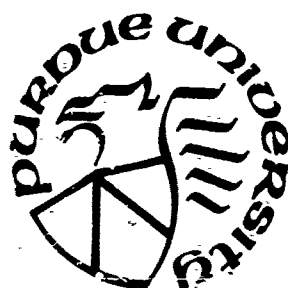
AD NO. _____

DDC FILE COPY

**SYNTACTIC ALGORITHMS
FOR IMAGE SEGMENTATION AND A
SPECIAL COMPUTER ARCHITECTURE
FOR IMAGE PROCESSING**

See 1473

**Janmin Keng
K. S. Fu**



**School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907**

BEST AVAILABLE COPY

Approved for public release;
distribution unlimited.

TR-EE 77-39

December 1977

This work was supported by AFOSR Grant No. 74-2661 and
ARPA Grant No. 0069-53-12855.

②

SYNTACTIC ALGORITHMS FOR IMAGE SEGMENTATION AND A SPECIAL COMPUTER

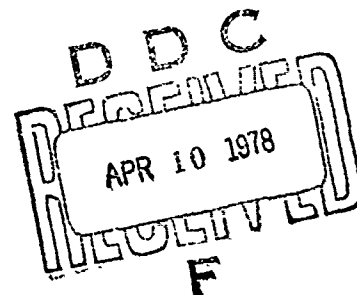
ARCHITECTURE FOR IMAGE PROCESSING

Janmin Keng and K. S. Fu

TR-EE 77-39

December 1977

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907



This work was supported by AFOSR Grant No. 74-2661 and ARPA Grant No. 0069-53-12855.

This document has been approved
for public release and sale; its
distribution is unlimited.

3.2.4	Experimental Results on Bridge and Commercial/Industrial Area Recognition from LANDSAT Images	101
3.3	Comparison of the Syntax-Directed and Syntax-Controlled Methods	101
4.	IMAGE SEGMENTATION USING A TREE GRAMMAR APPROACH.	106
4.1	Syntactic Image Segmentation Algorithm	108
4.1.1	Inference of Tree Grammars.	108
4.1.2	Texture Region Primitive Extraction	124
4.1.3	Boundary Primitive Extraction	129
4.1.4	Tree Grammar Analysis	138
4.2	Computer Experimental Results on Image Segmentation from LANDSAT Images	142
4.3	Computer Experimental Results of Image Segmentation from FLIR (Forward Looking Infrared) Images	145
4.3.1	Data Acquisition System of Infrared Images.	150
4.3.2	Experimental Results.	151
4.4	Summary.	166
5.	DESIGN FOR A SPECIAL COMPUTER ARCHITECTURE FOR IMAGE PROCESSING.	168
5.1	Previous Work and Comments	170
5.2	Physical Organization and Control Flow of the Proposed Computer Architecture	191
5.2.1	Physical Organization of the Proposed Computer Architecture	192
5.2.2	Control Flow of the Proposed Computer Architecture.	200
5.3	Analysis	204
5.3.1	Performance and Cost-Effectiveness Tradeoffs	204
5.3.2	Implementation of Statistical Methods	205
5.3.3	Syntactic Methods and Parallel Processing	208
5.4	Summary and Remarks.	217
6.	CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK.	219
6.1	Summary and Conclusions.	219
6.2	Suggestions for Further Work	222
	BIBLIOGRAPHY.	223
	APPENDICES	
	Appendix A. Sample Patterns for Inference of the Tree Grammar for Image Segmentation.	232

	Page
Appendix B. The Flow Chart of Highway Recognition Via the Syntax-Directed Method.	234
Appendix C. The Flow Chart of River Recognition Via the Syntax-Directed Method.	235
Appendix D. The Flow Chart of Bridge Detection Via the Syntax-Directed Method with Semantic Process.	236
Appendix E. Training Samples for Highway Grammar.	237
Appendix F. Sample Patterns for Inference of Tree Grammar for Military Vehicle.	241
Appendix G. The Flow Chart of Tree Grammar Inference Procedure	243
Appendix H. The Flow Chart of Tree Grammar Analysis	244
Appendix I. The Example of Tree Grammar Construction.	246

ACCESSION for	
NTIS	Write Section <input checked="" type="checkbox"/>
DDC	B.H. Section <input type="checkbox"/>
UNANNOUNCED	
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY NOTES	
Dist.	QUAL
A	

LIST OF TABLES

Table	Page
3.1 CPU time performance comparison of syntax-directed and syntax-controlled methods.	103

LIST OF FIGURES

Figure		Page
2.1	Block diagram of syntactic pattern recognition system.	23
2.2	General model of grammatical inference process	25
3.1	Flowchart for an image recognition system	32
3.2	A set of template matching 8x8 window patterns.	40
3.3(a)	Satellite image of northwest part of Indianapolis, Indiana	46
3.3(b)	Intermediate output after line smoothing process on Figure 3.3(a).	47
3.3(c)	Highway recognition result from Figure 3.3(a) by syntax-directed method	48
3.3(d)	City map of Indianapolis, Indiana	49
3.4(a)	Satellite image of downtown Chicago, Illinois	50
3.4(b)	Highway recognition result from Figure 3.4(a) by syntax-directed method	51
3.4(c)	Map of downtown Chicago, Illinois	52
3.5(a)	Satellite image of Harvey, Illinois	53
3.5(b)	Highway recognition result from Figure 3.5(a) by syntax-directed method	54
3.5(c)	City map of Harvey, Illinois.	55
3.6(a)	Satellite image of north part of San Francisco Bay area, California.	56
3.6(b)	River recognition result from Figure 3.6(a) by syntax-directed method	57

Figure		Page
3.6(c)	Topographic map of the same area of Figure 3.6(a)	58
3.7(a)	Satellite image of Lafayette area, Indiana (192x192 pixels)	59
3.7(b)	River recognition result from Figure 3.7(a) by syntax-directed method.	60
3.7(c)	City map of Lafayette, Indiana	61
3.8(a)	Topographic map of lower part of Figure 3.3(a)	64
3.8(b)	Bridge recognition result from Figure 3.3(a)	65
3.9(a)	Satellite image of Lafayette area, Indiana	66
3.9(b)	Bridge recognition result of Figure 3.9(a)	67
3.10(a)	Satellite image of northwest part of Indianapolis, Indiana area (96x96 pixels).	68
3.10(b)	Topographic map of same area as Figure 3.10(a)	69
3.10(c)	Intermediate output after preprocess of Figure 3.10(a).	70
3.10(d)	Highway recognition result on Figure 3.10(a)	71
3.10(e)	Intermediate output after exclusive or operation (EXOR) of the postprocessor.	72
3.10(f)	Commercial/industrial area recognition result.	73
3.10(g)	Urban development information extraction result of Figure 3.10(a).	74
3.11(a)	Commercial/industrial area recognition result from Figure 3.7(a)	75
3.11(b)	Urban development information extraction result of Figure 3.7(a)	76
3.12	Highway recognition result by the syntax-controlled method on Figure 3.3(a)	94
3.13	River recognition result by syntax-controlled method on Figure 3.3(a).	95

Figure		Page
3.14	Bridge recognition result by syntax-controlled method on Figure 3.3(a)	96
3.15(a)	Satellite image of downtown Indianapolis, Indiana	97
3.15(b)	Commercial/industrial area recognition by syntax-controlled method on Figure 3.15(a)	98
3.15(c)	Urban development information extraction result by syntax-controlled method on Figure 3.15(a)	99
3.15(d)	Indianapolis downtown map (marked area corresponds to Figure 3.15(a)	100
4.1	Block diagram of the system for image segmentation.	107
4.2	An illustrative example of variability texture measurement on an 11x11 image window.	127
4.3	Histogram of variability texture measurements	130
4.4	Histogram of angular second moment texture measurements.	131
4.5	Histogram of contrast texture measurements.	132
4.6	Histogram of correlation texture measurements	133
4.7	Satellite image of Indiana area	146
4.8(a)	Intermediate result after the texture region primitive extraction of the syntactic image segmentation algorithm on Figure 4.7.	147
4.8(b)	Result of syntactic image segmentation of Figure 4.7.	148
4.9	Point-by-point classification result of Figure 4.7.	149
4.10(a)	Infrared image of a tactical target scene	155
4.10(b)	Image segmentation result by the syntactic method on Figure 4.10(a).	155
4.10(c)	Image segment 3x3 of the background of Figure 4.10(a).	156
4.10(d)	Texture region primitive measurement result of Figure 4.10(c)	156

Figure	Page
4.10(e) Image segment 8x8 of the target area of Figure 4.10(a)	156
4.10(f) Texture region primitive measurement result of Figure 4.10(e)	156
4.11(a) Infrared image of a tactical target scene.	157
4.11(b) Image segmentation result by the syntactic method on Figure 4.11(a)	157
4.11(c) Image segment 6x6 of the background of Figure 4.11(a)	158
4.11(d) Texture region primitive measurement result of Figure 4.11(e)	158
4.11(e) Image segment 8x8 of the target area of Figure 4.11(a)	158
4.11(f) Texture region primitive measurement result of Figure 4.11(e)	158
4.12(a) Infrared image of a tactical target scene.	159
4.12(b) Image segmentation result by the syntactic method on Figure 4.12(a)	159
4.13(a) Infrared image of a tactical target scene.	160
4.13(b) Image segmentation result by the syntactic method on Figure 4.13(a)	160
4.14(a) Infrared image of a tactical target scene.	161
4.14(b) Image segmentation result by the syntactic method on Figure 4.14(a)	161
4.15(a) Infrared image of a tactical target scene.	162
4.15(b) Image segmentation result by the syntactic method on Figure 4.15(a)	162
4.16(a) Infrared image of a tactical target scene.	163
4.16(b) Image segmentation result by the syntactic method on Figure 4.16(a)	163
4.17(a) Infrared image of a tactical target scene.	164
4.17(b) Image segmentation result by the syntactic method on Figure 4.17(a)	164

Figure	Page
4.18(a) Infrared image of a tactical target scene.	165
4.18(b) Image segmentation result by the syntactic method on Figure 4.18(a)	165
5.1 Schematic diagram of stalactite of pattern articulation unit in ILLIAC III computer	172
5.2 Block diagram of Illinois pattern recognition computer ILLIAC III.	173
5.3 Block diagram of computer system in Advanced Automation Research Laboratory	175
5.4 CLIP 3 cell logic	177
5.5 Schematic diagram of hybrid CLIP 3 system.	179
5.6 CLIP 4 cell logic diagram.	180
5.7(a) Block diagram of parallel picture processing machine.	182
5.7(b) Picture registers.	182
5.7(c) Neighborhood matching logic.	182
5.7(d) Block diagram of PICAP computer system	182
5.8 Block diagram of flexible processor by Control Data Corporation	185
5.9 Block diagram of TOSPICS computer.	186
5.10 Block diagram of STARAN computer	189
5.11 Control units of designed computer architecture for image processing	195
5.12 Parallel processor of designed computer architecture	196
5.13 Sequential arithmetic processor of designed computer architecture.	199
5.14 Memory hierarchy of designed computer architecture	201
5.15 Control flow of designed computer architecture for image processing, array processing	202
5.16 Performance improvement versus increments of number of processor.	206

Figure	Page
5.17	Cost estimation versus number of processors. 207
5.18	Parallel tree parsing procedure. 214
5.19	Conventional tree parsing procedure. 215

ABSTRACT

Several efficient algorithms for image recognition and segmentation and a new computer architecture for image processing are proposed. The algorithms are "syntactic" in that they perform structural or spatial analysis rather than statistical analysis, and a "grammar" is inferred for describing the structures of patterns in an image. Depending on the requirements of the problem, an appropriate grammatical approach is used by the syntactic algorithm.

A finite-state string grammar is applied to the image recognition of highways, rivers, bridges, and commercial/industrial areas from LANDSAT images. There are two major methods in the string grammar approach for image recognition; namely, the syntax-directed method and syntax-controlled method. For the syntax-directed method, syntactic analysis is performed by a template matching which is directed by the syntactic rules. For the syntax-controlled method an automaton which is directly controlled by the syntactic rules is used for the syntactic analysis.

A tree grammar is applied to the image segmentation of terrain and tactical targets from LANDSAT and infrared images respectively. The tree grammar approach utilizes a tree automaton to extract the boundaries of the homogeneous region segments of the image. The homogeneity of the region segment is obtained through texture feature measurements of the image.

The computer architecture proposed is a special purpose system in that it can perform an image processing task on several picture-points of an image at the same time, and thus takes advantage of the fact that image processing tasks usually exhibit "parallelism". This architecture uses a distributed computing approach. Two major features are the re-configurable capability, and the method of computer exploitation of task parallelism. Finally, a parallel parsing scheme for tree grammar is used to demonstrate the higher efficiency of the proposed computer architecture than the conventional parsing scheme.

CHAPTER 1

INTRODUCTION

Image segmentation is a computer technique that breaks an image into different regions, each having homogeneous properties [83]. In image analysis or scene analysis the desired result is a computer-generated description of the given image or scene. The computer-generated description refers to specific parts (regions or objects) in the image or scene. Therefore, a first step is to divide the image into these parts; that is, to do image segmentation. Image segmentation is also an important stage in sample classification [1], and image compression [83].

An algorithm is a computational procedure showing the steps the computer is asked to perform [113]. Algorithms involving statistical operations are statistical algorithms. Algorithms which perform structural or spatial analysis in image processing problems are called "syntactic" algorithms because of the analogy between the structure of patterns and the syntax of languages [2].

The "language" that provides the structural description of patterns in terms of a set of pattern primitives and their composition operations is called the "pattern-description language." The rules governing the composition of primitives into patterns are usually specified by the so called "grammar" of the pattern description language. After each primitive within the pattern is identified, the recognition process is accomplished by performing a syntactic analysis of the "sentence"

describing the given pattern to determine whether or not it is syntactically (or grammatically) correct with respect to the specified grammar. There are four types of grammars [2] according to form; namely, type 0 (unrestricted) grammars, type 1 (context-sensitive) grammars, type 2 (context-free) grammars, and type 3 (finite-state) grammars. "Finite state" refers to the fact that these grammars have various control states (a finite number of them). There are two types of finite state grammars: string grammars and tree grammars [13]. If a finite state grammar is used in the syntax analysis, the analysis is called a finite state grammar approach. The selection of an appropriate grammar for image processing usually depends on the requirements of the problem. A syntactic algorithm for image segmentation and two for image recognition are presented in Chapters 3 and 4.

An algorithm can be implemented by computer software (programs) or hardware. If it is considered that the potential usefulness of a specific algorithm justifies the time and cost, specific hardware can be built to perform the algorithm. That is, a special small computer can be built to perform just that one task. With the ever increasing interest in, and useful applications of image processing, a consideration of building a special purpose computer primarily for image processing is justified. Image processing done on large computers takes a great deal of memory space and is very time consuming. Thus, the cost of the computer time is high. So this cost has to be weighed against the cost of building a special purpose computer.

Image processing tasks usually exhibit "parallelism." That is, hardware can be built to perform a task on several picture-points of an image at the same time, i.e., in parallel. This makes image processing

as the method of segmenting an image. They involve region merging, region dividing, or a combination of merging and dividing.

a. Region Merging

This has been done by five approaches: statistical, linguistic, decision-theoretical, relaxation, and interpretation guided.

- (i) The statistical approach to region merging has been developed by Brice and Fennan [34]; Bajcsy [39]; Gupta and Wintz [40]; and Jarvis [114];
- (ii) the linguistic approach has been investigated by Tsuji and Fujiwara [41];
- (iii) the decision-theoretical approach was developed by Yakimovsky and Feldman [42];
- (iv) the relaxation approach was developed by Rosenfeld, Hummel, and Zucker [78]; and
- (v) the interpretation guided approach has been investigated by Tenenbaum and Barrow [79].

b. Region Dividing

This involves successively partitioning the image by certain criteria and was first put forth by Robertson [44], and Klinger [45].

c. Combination

A combination of merging and dividing has been proposed by Horowitz and Pavlidis [46,71].

3) "Edge" Detection Methods: These methods define as "edges" the boundary between two different objects in a picture, (for example, the "edge" between a human neck and a sweater neck) and consider these edges as the boundaries of the segments of the image. There are several approaches as follows:

a. Template Matching

This approach has been explored by Griffith [47,48]; and Hueckel [49,50];

b. Gradient Operator

This approach has been studied by Duda and Hart [57]; Rosenfeld [61]; Rosenfeld and Thurston [58,59]; Rosenfeld and Troy [68]; Rosenfeld and Thomas [60]; Persoon [67]; Wechsler [112]; and Thompson [85];

c. Boundary-Search

This approach investigated by Rosenfeld [52]; and Kelley [53]; and

d. Line-Fitting

This approach developed by Hough [54]; Duda and Hart [55]; and Pavlidis [56].

Most of the above techniques are statistical. None of them are based on the syntactic approach. None of them have explicitly used structural and contextual information. These statistical techniques suffer from costly computer processing time. An image often exhibits a hierarchical structure. Therefore, image segmentation can be approached by the syntactic method. A syntactic method based on the finite state (string) grammar approach has been developed for image recognition in Chapter 3. A syntactic method based on the tree grammar approach has been developed for image segmentation in Chapter 4. It is desirable to make syntactic algorithms useful in real-world applications. Thus, these syntactic algorithms have all been tested on real-world data such as satellite images, aerophotographic images, and infrared images. The experimental computer results show that these syntactic algorithms are useful for image recognition and segmentation.

1.2 DESIGN FOR A SPECIAL COMPUTER ARCHITECTURE FOR IMAGE PROCESSING

Previous designs for special computer architecture for image processing basically fall into two categories: bit-plane processing and distributed processing.

a. Bit-Plane Processing

The bit-plane processing approach performs the arithmetic computation on image points which are stored in Boolean bit planes. The special-purpose computers developed by this approach are the Illinois pattern recognition computer (ILLIAC III) [87,100], the Digital Parallel Processor (DPP) [90,91], the Cellular Logic Image Processor (CLIP 4) [92], and the Parallel Picture Processing Machine (PPM) [88,89].

b. Distributed Processing

In the distributed computing approach, the configuration of the processors forms an architecture such that the computational load is shared by these processors through software and/or hardware control. The special purpose computers that have been built utilizing this approach are the Flexible Processor by Control Data Corporation [97], Toshiba Picture Processing System (TOSPICS) [93,116] by Toshiba Corporation, and STARAN computer by Goodyear Aerospace Corporation [108,109].

A weakness of all the above special computers for image processing is that the computer systems are not reconfigurable, that is, the computer can only operate in one of the four modes: SISD (single instruction stream single data stream), MISD (multiple instruction stream single data stream), SIMD (single instruction stream multiple data stream), or MIMD (multiple instruction stream multiple data stream). Because of the great variety of sensor types, and the many applications, image processing algorithms require that the computer system be reconfigurable. Therefore,

a design for a special computer for image processing which is reconfigurable is proposed in Chapter 5.

1.3 RESEARCH SUMMARY

This research deals with three areas: syntactic pattern recognition; information extraction and image understanding; and special computer architecture for image processing.

In the area of syntactic pattern recognition, the finite-state string grammar approach used for image recognition is presented in Chapter 3. Two syntactic methods have been developed. One is a syntax-directed method and the other is a syntax-controlled method. The syntax-directed method uses a set of templates as a recognizer. A deterministic finite-state automaton is used as a recognizer in the syntax-controlled method. An interactive grammatical inference procedure was devised for the syntax-directed method. A k-tail finite-state grammatical inference procedure [20], which is a procedure to minimize the number of states by equivalence partitioning based on the length "k" of the derivatives of the sample patterns, is used in a fully computer automated procedure for the syntax controlled method. A comparative study on the syntax-directed (commonly known as template matching) and syntax-controlled methods were undertaken. This led to a decision in favor of the syntax-controlled method. The performances of the recognition by the different finite state grammars, which are inferred by the k-tail inference procedure with different values of k, were studied. This revealed that the grammar inferred by a high value of k is more precise in characterizing the syntactic patterns than that inferred by a low value of k, but the computer processing time for the recognition by the corresponding finite state automaton increases. These syntactic methods have been implemented and applied to the

recognition of highways, rivers, bridges, and commercial/industrial areas from satellite images. The finite-state string grammar has been shown to be able to characterize the structure of highways and rivers. The computer results were accurate even though the resolution of the input image was low due to being collected by satellite at a very high altitude (approximately 570 miles) [82].

In the area of information extraction and image understanding, the objective of the research was to achieve a better understanding of image structure and to use this knowledge to develop a technique for image analysis and automatic information extraction. The results obtained from applying syntactic pattern recognition to satellite images of highways, rivers, bridges, and commercial/industrial areas are useful for image analysis and relevant to military applications. The automated methods of extracting such information as position coordinates and lengths of bridges and centers and sizes of a commercial/industrial areas as presented in Chapter 3 provides a high level understanding of imagery by computer automation. The syntactic image segmentation algorithm presented in Chapter 4 incorporates textural discrimination and boundary structure analysis. The tree grammar approach is applied. A tree transformational grammar and its inference procedure is introduced to reduce the noise and irregularities in patterns. The syntactic image segmentation algorithm was applied to tactical target detection from infrared images.

In the area of special computer architecture for image processing, several previously proposed special computers for image processing were reviewed. The proposed special computer architecture in Chapter 5 was designed using a distributed computing approach. This computer is comprised of a Parallel Processor (PP) and a Sequential Arithmetic

Processor (SAP). There are two major features new to the field of special computer architecture for image processing. They are a reconfigurable capability, and a special method of parallelism. These contribute a high flexibility and a high performance capability for image processing to the proposed computer architecture. The reconfigurable capability enables the proposed computer to satisfy the large variety of applications in image processing. This capability is obtained through the Control Unit of Parallel Processor (CUPP) which reconfigures the parallel processor between the SIMD mode and the MIMD mode. The parallelism of the task is exploited by the parallel processor to obtain high speed performance. At the same time the operations of the sequential arithmetic processor are pipelined to the parallel processor under program control in those tasks which can be decomposed into pipeline processing. Therefore, the exploitation of parallelism results in parallelism and pipelining simultaneously. Thus, the computer architecture achieves high flexibility and performance.

CHAPTER 2

PREVIOUS WORK

2.1 PREVIOUS RESEARCH WORK IN IMAGE SEGMENTATION

As mentioned in Chapter 1, previous research in the field of image segmentation falls into three major categories; characteristics thresholding, region extraction, and edge detection.

2.1.1 Characteristics Thresholding Methods

The whole field of image processing has become possible because of the development of the ability to "digitize" a picture or image; that is, the picture is digitized into a matrix of n by m pixels (picture elements). Then, each pixel is given a grey level value corresponding to the amount of light it transmits. These grey levels lend themselves to mathematical manipulation, and functions can be written involving them. The first task is to find characteristic functions of grey levels. The second is to find a threshold in such a function that will make a significant division in the type of pixels. There are two basic approaches to thresholding the characteristic function of grey levels for image segmentation; the statistical and the structural.

a. Statistical Approach

In this approach, a picture is divided into different regions by thresholding the value of an appropriate local picture property. For example, all physical objects have a constant reflectance over their

surfaces. Therefore, an object of uniform grey level is considered to be a homogeneous object. Zucker, Rosenfeld, and Davis [26] proposed that a good method of segmenting a picture into regions of uniform grey levels would be to examine the histogram of the grey levels in the picture. Instead of measuring the grey level of each picture point, a set of spot detectors having a range of sizes is used to provide local property values upon which to base the segmentation. There is one problem in this technique; that is, that due to overlap of the spot detector in the measurement of each point, sometimes the histogram valley, necessary to threshold the picture could not be found. So non-maximum feature values were suppressed over an area corresponding to the receptive field of each spot detector; that is, a spot value was ignored if a larger value existed at some point in its receptive field. The threshold for segmentation was selected as the lowest point between two peaks in the histogram. But there are only limited results on bimodal cases in Zucker, Rosenfeld, and Davis's [26] paper. Further complicated experiments, such as the multimodal cases should be studied in order to show the generality of this technique.

In some cases, where the black and white dots occur in both the significant regions and in the background (e.g., in the cases where the probability of occurrence of a black dot is 0.6 in the significant region and 0.4 in background), the method in [26] will not work because the threshold cannot be found. Davis, Rosenfeld, and Weszka [29] applied local averaging of grey level values to every point of the picture. Then the histogram was built by the average grey levels of the picture. But the method will not work if the picture also contains other adjacent regions in which the average grey levels are higher and

lower than that of the initial region. Weszka, Nagel, and Rosenfeld [30] applied the Laplacian operator to determine points that lie on or near the edges of objects to try to get around this problem. In working with a grey level histogram, the peaks are sometimes very unequal in size and the valley is broad. Here, the thresholding is difficult.

Ohlander [68] used multiple sources of data and the threshold operation based on histograms. Thus, there exists the option of examining nine histograms from these sources to determine the most sharply defined feature. Thresholding on limits provided by the minima bounding the best peak in the histogram will furnish clusters of points which are uniform for the given feature. Then these regions are extracted. But, the number of sensory parameters and variety of picture operations require, for this system, large amounts of storage space and heavy expenditures of computational time. In this system, the control of parameters is determined by human interaction. Even with human interaction and 9 hours of CPU time, the heavy input and output requirements increases the real time processing to 18 hours or more to process a scene of 600x800 points on a PDP 10 computer. And as the amount of noise in the information becomes greater and greater, the resolution of this system becomes less and less.

Carlton and Mitchell [86] used texture and grey level information for image segmentation. This technique uses a texture measure that counts the number of local extrema in a window centered at each pixel. This results in an intermediate grey level picture representation of a texture property. These intermediate pictures are used to derive starting points in each region to be segmented. The segmentation is

completed by assigning each pixel to a starting point using a distance criteria. For this technique, there are several thresholds which must be set to make it operate; namely, extrema size, window sizes, and distance criteria. If the input data is not quite homogeneous, the optimal set of thresholds needs to be found.

Wall, Klinger and Castleman [27] and Wall [28] proposed five models to represent objects in image analysis and their corresponding histograms. The five models are the ideal object image function, the truncated wedge function, the circular Gaussian function, the Gaussian through function, and the Gaussian edge function. The ideal object image function $I(x,y)$ is a function having constant grey level I_m , inside an arbitrary set, S , and zero outside. S is a simple, connected subset of the region with a boundary of arbitrary shape. If $(x,y) \in S$, $I(x,y) = I_m$; if $(x,y) \notin S$, $I(x,y) = 0$. The truncated wedge function is a function having constant grey level, I_m , and slope, b , from the constant value of grey level to zero value of grey level. The circular Gaussian function of object image is defined by

$$I(x,y) = I_m e^{-[(x-u_x)^2 + (y-u_y)^2]/2\sigma^2} ; (x,y) \in d$$

$$I(x,y) = 0 ; (x,y) \notin d$$

where $d = \{(x,y) \in R; \frac{(x-u_x)^2}{2\sigma^2} + \frac{(y-u_y)^2}{2\sigma^2} \leq Z\}$, u_x, u_y, σ are arbitrary constants and $Z \geq 3$. The Gaussian through function is defined by

$$I(x,y) = I_m e^{-(x-u_x)^2/2\sigma^2} ; (x,y) \in r$$

$$I(x,y) = 0 ; (x,y) \notin r$$

where $r = \{(x,y) \in R; (U_y - \frac{L}{2} < y < U_y + \frac{L}{2}), \frac{(x-U_x)^2}{2\sigma^2} \leq Z\}$ U_x , U_y , σ , and

L are arbitrary constants and $Z \geq 3\sigma$. The Gaussian edge function is generated by placing one-half of a circular Gaussian image function on each end of the Gaussian through function.

For the extension of grey level, Yachida and Tsuji [33] found uniform color regions by utilizing color information. So, color information is helpful in image segmentation.

b. Structural Approach

In the structural approach to characteristics thresholding, the uniformity of complex image properties such as size, shape, or arrangement of subpatterns in a picture is examined.

Tsuji and Tomita [32], and Tomita and Yachida [31] described a method for dividing the input scene into regions by thresholding the histograms of the grey level values of several structural descriptors. For the purpose of saving memory space and computing time, several specific descriptors were selected. They are shape, size, position, and density. The size descriptor specifies the area and perimeter of each unit region. This position descriptor gives the two dimensional coordinates of the center of gravity of the unit region. The shape descriptor is selected in such a way that it gives rough information about the shape and is not sensitive to sampling noise in the process of digitizing the picture. A region A in a given set, s , has a density descriptor, D_s , whose value is the minimum distance from A to other regions in s . The partitioning procedure works as following: first, the picture descriptors are evaluated and their list is constructed.

Second, the histograms of the values of the descriptors are computed and a filtering technique smooths the small peaks that lie close to each other. Third, a supervisor selects the most promising descriptors for classifying the regions into groups. The most promising descriptors are those descriptors whose histograms have deep valleys. The supervisor puts the thresholds at the bottoms of the valleys. Fourth, the density descriptors of the members of each group are evaluated. If some members have density descriptors whose values are larger than a predetermined threshold value, they are excluded from the groups as isolated regions. Finally, a boundary test is done to check whether or not any regions in the group of isolated elements are located between two regions or surrounded by a region. They are merged with the region if only one region touches them. Thus, the structural analysis is finished and the goal, partition of the picture, is achieved.

2.1.2 Region Extraction Methods

These methods utilize extraction of regions to segment the picture. There are three approaches to region extraction: merging, dividing, a combination of merging and dividing.

a. Region Merging Approach

The region merging approach begins with a well formed partition (e.g., the picture consists of n^2 square pixels and the pixel is $|x|$ in size), and processes it by merging adjacent elements together that are found to be similar in certain characteristics. There are five types of region merging: statistical, linguistic, decision-theoretic, "relaxation", and interpretation-guided.

(i) Statistical Region Merging

Here, the merging criteria are based on certain statistical characteristics. Brice and Fennema [34] use unit regions as basic data and process them by successive merging of the unit region toward a final image partition. Similar approaches used by Strong and Rosenfeld [35], Harlow and Eisenbers [36], and Rodd [37]. Bajcsy [39] uses two types of merging of logic sequences; from left to right, and from right to left. Kettig [38] uses a similar method on multispectral remotely sensed imagery data. His experimental results show that this approach suffers from slow computer processing time. Gupta and Wintz [40] proposed a merging sequence which expands horizontally and vertically by absorbing more and more pixels until it reaches nature boundaries.

(ii) Linguistic Region Merging

Since most grammars are useful in analyzing a one dimensional string, Tsuji and Fujiwara [41] proposed applying sequentially two grammars of one dimensional strings for picture segmentation. The picture primitives are line, curve, edge or undefined segments. The system works as follows: the first stage of processing is the search for line segments. Then a line fitter tries to connect the line segments. The horizontal grammar is constructed manually in that a horizontal parser gives a label to each segment, and examines the horizontal contexts in order to join several segments into a new longer segment. The results are the set of picture sentences of horizontal scan. Based on the concept of coupling, a set of grammar rules has been written as vertical grammar. The vertical parser analyzes the vertical contexts of symbols in picture segments and generates region sentences which

give the characteristics of all surfaces in the picture. The final step of vertical parsing is to combine one region with another. This approach works fairly well on some artificial block type of images, but fails in classifying scenes that have high lights, shadows, and different textures.

(iii) Decision-Theoretic Region Merging

This approach applies Bayesian decision rule techniques and uses problem-dependent information (semantics) to solve the picture segmentation problem. Yakimovsky and Feldman [42] and Yakimovsky [43] present a theoretical framework for a system which incorporates neighborhood information in a region analyzer. The two central ideas are the use of a utility function to measure the value of various alternatives, and an optimality theorem. A disadvantage in the approach is the assumption that the interpretation of a region depends only upon adjacent regions. The choice of local measurements around each point is, of course, a crucial factor but interpretation can sometimes depend on a region quite far away. Also it seems that the computer's "learning" procedure needs to be refined in this approach.

(iv) "Relaxation" Region Merging

Another approach to region merging is called the "relaxation" process proposed by Rosenfeld, et al [78]. This process has been applied to scene labeling, line enhancement [80], and template matching. The relaxation process, also called iterative probabilistic process, first estimates for each point, P , the probability, P_i , that it belongs to each of the possible classes. Then the P_i is increased if supporting evidence is found for it or decreased where contradictory evidence is

found. When for a given point, P , only one class has a high probability, then it is classified as that class. In this approach, the estimated initial probability, P_i , for each point would effect the converging time for iteration. The computer processing time can be costly for this method because of the large number of iteration needed.

(v) Interpretation-Guided Region Merging

Tenenbaum and Barrow [79] proposed the IGS (Interpretation Guided Segmentation) approach to region merging. This technique relates human knowledge to a certain set of rules and iteratively processes these rules until it achieves a final segmentation result. Finding a set of rules which effectively describes the knowledge of the scene is important. Because the rules have to be iteratively processed, the computer processing time is large.

b. Region Dividing Approach

The region dividing approach begins with dividing the picture into several parts then subdividing each part until the partitioning satisfies certain stopping criteria. Robertson [44] designed his partition algorithm on the assumption that a region contains a boundary. The homogeneous property that he considers is the mean vector of brightness functions of the multispectral remotely sensed image.

Klinger [45] suggests a partition idea through the refinement of objects into four equal quadrants such as, northwest, northeast, southwest, and southeast quadrants. He also [69] applies a regular decomposition to divide the picture area into successively smaller quadrants. The concept of regular decomposition is, first to represent a digitized picture consisting of spatial subsets of different sizes marked either

"informative for scene description" or "non-informative" and second, to discard picture elements (pixels) that belong to "non-informative" subsets. Initially, the entire digitized picture is a quadrant. If nothing informative is contained, such a quadrant may be entirely eliminated from the data structure. If a large amount of information is found in the quadrant, the quadrant should be saved. If the decomposition algorithm fails to make a decision about a picture quadrant, it is subdivided and each of the four quadrants is processed by the same procedure. Thus, the regular decomposition is a process of searching for picture areas where there is "informative" data present. This method is used to build computer-searchable data representation of an image.

c. Combination of Merging and Dividing

This approach to region extraction starts from an arbitrary partition and dynamically determines when to merge or divide. Horowitz and Pavlidis [46] used this idea in devising their image segmentation by a directed split-and-merge procedure. They have also refined their algorithm by applying a grouping algorithm which combines two adjacent regions, provided these initial regions satisfy a given property. It eliminates small regions which are due to noise or to transitions between large regions. For the purpose of identifying multiple connected regions corresponding to cut notes in the picture, Horowitz and Pavlidis [70] used graph analysis to identify these regions. But the a priori information of the given property that they use is the drawback to their algorithms.

2.1.3 Edge Detection Methods

Another method for image segmentation is finding the boundaries between regions in order to extract the homogeneous regions. There are several approaches; such as, template matching, gradient operator, boundary searching and line fitting.

a. Template Matching

Template matching can be done in a decision-theoretic way. The decision problem is to compute the probability that a line representing a real edge is centered in some long narrow area. Griffith [47,48,57] discusses the optimal use of intensity information to detect edges in a block world. Hueckel [49] proposes a method by asking what edge element will best fit the intensities in a given region. An extension of the foregoing techniques to detect line and edge-line is presented in Hueckel [50]. This approach is capable of telling the exact orientation of the line segment.

b. Boundary Operator

The gradient boundary operator is defined as

$$\nabla g(x,y) = \frac{\partial g}{\partial x} \vec{\lambda}_x + \frac{\partial g}{\partial y} \vec{\lambda}_y$$

$$\text{and } |\nabla g(x,y)| = [(\frac{\partial g}{\partial x})^2 + (\frac{\partial g}{\partial y})^2]^{1/2}.$$

If the picture is noisy, then smoothing techniques should be employed before applying the gradient operator according to [57]. Rosenfeld [61] indicates that when there is noise incorporated into the smoothing procedure, the larger the size of the neighborhood, the less precisely is the edge located. Procedures for detecting abrupt changes in average grey level value in order to locate the edges, are presented in Rosenfeld and Thurston [58,59]. Rosenfeld and Troy [66] and Rosenfeld and

Thomas [60] apply this concept to detect the texture edges where the two regions differ with respect to the average value of some local properties. Recently, Persoon [67] devised a new gradient operator algorithm for edge detection performing experiments on rib extraction from chest x-ray pictures.

In March of 1977, Thompson [85] applied the modified Roberts cross operator for textural boundary detection. The Roberts cross operator is defined as $R(i,j) = |P(i,j) - P(i+1,j+1)| + |P(i+1,j) - P(i,j+1)|$. In [85], the textural boundary operator is proposed as $T = D(a,d) + D(b,c)$. $D(a,d)$ is the computed texture dissimilarity between region a and d. Regions a, b, c and d are the quadrants of northwest, northeast, southwest, and southeast respectively. The edge finding method consists of two stages. First, a map is produced by applying the textural boundary operator to selected points in an image. A second edge map is then provided by smearing each point in the first map along the direction of edge orientation.

c. Boundary Searching

This approach detects the boundary by means of applying an operator to search the picture to locate the boundary between regions. One of the techniques called "the contour following technique" is an application of the idea that previous knowledge about the existence of an edge from the operator might give us a good prediction of the location of the next edge element [52]. In [53], the planning strategy was applied to the contour following search. The technique of contour following with planning strategy should be quite good, as it applies knowledge from previous small areas. But it fails when the parts of the picture are independent and uncorrelated.

d. Line Fitting

Hough [54] suggested the use of a set of parameters to fit a group of points with line segments. Duda and Hart [55] indicated that the use of angle-radius rather than slope-intercept parameters will simplify the computation in Hough's work. Pavlidis [56] proposed numerical functional approximation to fit the boundary. Several algorithms of boundary detection [62,63,64,65] are of this category. These techniques can contribute to the segmentation of a picture. But, sometimes structure of the boundary of a picture is complicated and the line fitting approach becomes inadequate.

2.2 PREVIOUS WORK IN SYNTACTIC PATTERN RECOGNITION

In syntactic pattern recognition, a "grammar" is used to characterize a set of patterns. The syntactic method has the power of describing and classifying patterns. For the description of a pattern, a pattern is represented by a sentence in a language which is specified by a grammar [2,5]. This language provides the structural description of patterns by a set of pattern primitives and their relational rules. The recognition of a pattern is accomplished by a syntax analysis according to the grammar. A block diagram of a syntactic pattern recognition system is shown in Figure 2.1. The upper part of the diagram is the recognition part and the lower part is the analysis part. For the purpose of describing and recognizing the pictorial patterns, various two-dimensional grammars called tree grammars, web grammars, and graph grammars have been developed and applied to syntactic pattern recognition. A good survey of the early work in the field of syntactic pattern recognition was written by Fu [2]. More recent surveys are found in [4,5,6,12].

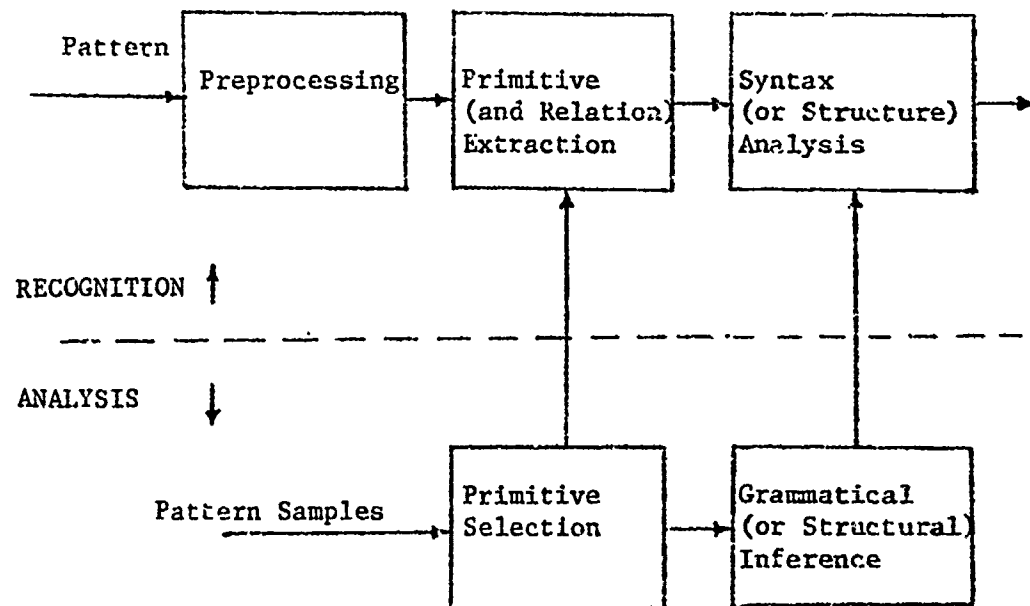


Fig. 2.1 Block diagram of syntactic pattern recognition system

Previous research work in this field falls into three major areas; namely; grammatical inference, syntactic analysis, and applications.

2.2.1 Grammatical Inference

Fu and Booth wrote an intensive survey on grammatical inference in [20]. The grammatical inference is part of the analysis for syntactic pattern recognition shown in Figure 2.1. In order to realistically describe a class of patterns under study, the grammar used in syntactic pattern recognition is hoped to be directly inferred from a set of sample patterns. Figure 2.2 shows the general model of the grammatical inference process [2]. A source generates sentences or patterns of form $x_i = a_{i_1}, \dots, a_{i_n}$ where the symbols a_i are elements from a finite set V_T called the set of terminal symbols. These sentences are assumed to possess some unique structural features which are characterized by a grammar, G , which can be used to model the source. All of the sentences which can be generated by the source are contained in the set $L(G)$, the language generated by G , while all of the sentences which cannot be generated are contained in the complement set $\overline{L(G)}$. An observer is given a finite set s^+ of sentences which are from $L(G)$ and another finite set s^- of sentences from $\overline{L(G)}$. Using this information the observer infers the syntactic rules of the unknown grammar G . The sentences which belong to s^+ are defined by the properties of G . These sentences from s^- are also input to the observer. In section 2.3 of this chapter, more details on grammatical inference algorithms are presented. A tree grammar inference algorithm is described, and an other grammatical inference algorithm for a tree transformational grammar is proposed.

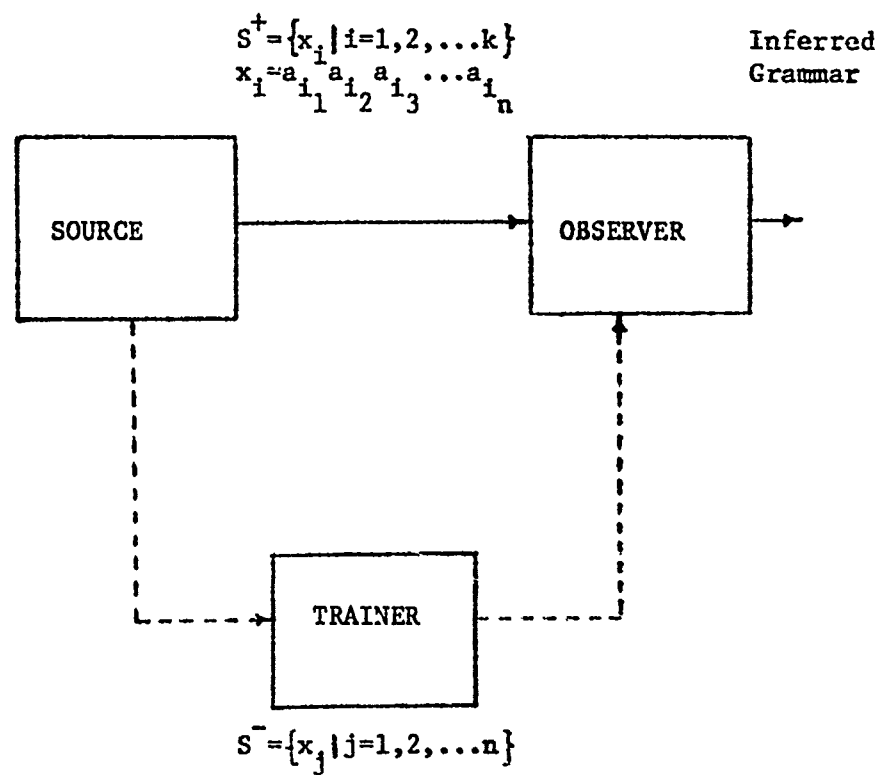


Fig. 2.2 General model of grammatical inference process

2.2.2 Syntactic Analysis

After a grammar is constructed to generate a language which would describe the patterns under study, the next step is to design a recognizer that will recognize the patterns according to the grammar. The recognizer designed for a particular grammar will recognize only the patterns in the class corresponding to this grammar. If the grammar is finite, the syntactic analysis is done by the finite state automata [2]. A recent development in this area is error-correcting syntactic analysis [16,19,22]. In some applications, a certain amount of uncertainty exists in the process. In order to describe noisy and distorted patterns, the use of transformational grammars, stochastic languages, and approximation have been suggested [2,5,23]. For the recognition of noisy and distorted patterns, error-correcting syntactic analyzers have been proposed. Fung and Fu [19,22] used the maximum likelihood decision criterion as decision rule to resolve ambiguities for stochastic language. The basic approach is to use the concept of transformations of strings. The error-correcting parser first induces error transformations into the original grammar to give a covering grammar whose language is universal. To search for the string that satisfies the decision rules, the parsing algorithm is a conventional parser with a provision added for bookkeeping of the number of transformations used. The probabilistic models and stochastic error-correcting parsing techniques were applied to the recognition of noisy patterns.

"Tree" system approach was first introduced to syntactic pattern recognition by Fu and Bhargava [13]. In [16], the error-correcting tree automata was proposed. Unlike the string case, where only the relation between symbols is left-right concatenation, a tree structure would

become deformed by deletion or insertion errors. The structure-preserved error-correcting tree automata takes only substitution errors into consideration. A generalized error-correcting tree automaton consists of five types of error transformations; namely, substitution, stretch, branch, splits and deletion. The distance between two trees is the least-cost sequence of error transformations needed to transform one tree to the other. Tree distance is measurable between trees of different structures. Based on the measurements, the error-correcting parser accepts structurally distorted as well as node-mislabeled trees such that their minimum distance corrections can be found [16].

For the purpose of increasing the flexibility of the syntactic method, Fu and Lu [17] proposed a clustering procedure for syntactic patterns. This procedure measures the distance between patterns and establishes a similarity measure. A similarity measure between two syntactic patterns includes the similarities of both their structures and primitives. A nearest-neighbor recognition rule is then applied with the similarity measures and a clustering algorithm is proposed for syntactic patterns. If the correct classifications of pattern samples are known, the proposed nearest neighbor recognition rule can be applied to determine the classification and structural description of an unknown pattern. When the correct classification of pattern samples is unknown, a non-supervised procedure must be used. In this case, the clustering procedure can still be applied. When using error-correcting parsers in cluster analysis, after the clustering result is obtained, only a conventional non-error-correcting parser needs to be implemented for recognition. The flexibility of the syntactic method is greatly improved by this clustering procedure.

2.2.3 Applications

Fu and Bhargava [2] applied the tree system approach to the analysis of Bubble Chamber films. The tree system approach utilizes a set of grammars to describe and recognize the Bubble Chamber patterns. Hoayer and Fu [11] applied the tree system approach which utilizes sets of grammars to describe and recognize fingerprint patterns. The tree system approach was also applied to LANDSAT data interpretation which analyzes the processed results from statistical methods to improve some of the data interpretation of LANDSAT images [15]. Brayer and Fu [14] and Fu [3,4] approached the LANDSAT data interpretation through "web" grammar analysis which utilizes web grammar to describe the contextual and spatial information. A web grammar model was developed and used to improve the accuracy of the classification and to find some new classes. Pavlidis [23,96] applied the grammars, which are characterized by a set of primitives corresponding to the case under study, to shape recognition. His applications are numeral and character recognition, and industrial circuit board defect detection.

2.3 COMMENTS ON THE PREVIOUS RESEARCH WORK

Compared with the techniques in the "characteristics" thresholding approach, the grey level histogram provides global knowledge about the segmentation of a picture. Concerning the computer processing time involved in image segmentation, the grey level histogram thresholding approach is quite fast because the mathematical operator manipulation is much simpler than gradient or Laplacian thresholding. But in the histogram thresholding approach, where there are small regions, they show up in the histogram as small peaks. Hence, most of the time the histogram thresh-

holding approach fails to segment these regions of the image. Another problem is the case in which two or more different objects have many overlapping parts in the histogram; here again the approach fails to segment the image. Therefore, techniques similar to the gradient operation, Laplacian operation or modified Laplacian operator have been developed by thresholding some local property, and Ohlander [68] tried using nine different types of sensory data for thresholding histograms. All these approaches suffer from heavy expenditures due to slow computer processing time. After this examination of the thresholding of histograms, we must consider that it is not the best approach for image segmentation.

When structures of patterns or textures of the image are complex, the characteristics thresholding method by structural approach is useful. But, the discrimination between the elementary subpatterns or grains of the image has to be assumed to be easily obtainable and this structural approach needs much computer processing time.

The region extraction method is attractive in two major approaches; the merging approach and the dividing approach. In comparing these two approaches, if the region merging approach is used every time two regions are merged together, the sample statistics can be calculated simply from the statistics of those two regions and thus the processing time is not so large. In the region dividing approach, every time a region is divided, new statistics must be calculated again causing large processing time. The combination of merging and dividing is considered a combinatory method, but the computer processing time is still very large.

In the edge detection methods for image segmentation, if boundaries between regions can be expressed by some definite form, then template matching is suggested. From these observations, there seems to be dif-

ferent appropriate approaches for different types of images. That is, it is difficult to find a general method for all images. Some of these techniques calculate second order statistics and some of them need planning strategy. However, all of these techniques require a great deal of computer processing time. In conclusion, we consider that there are problems associated with all of the previous methods. Therefore in Chapter 4, the syntactic image segmentation method is proposed. Before presenting the image segmentation by tree grammar (high dimensional) in Chapter 4, the image recognition by finite state string grammar (one dimensional) will be presented in Chapter 3.

CHAPTER 3

IMAGE RECOGNITION BY A STRING GRAMMAR APPROACH

The syntactic approach to image recognition has three major parts (Figure 3.1): Preprocessor, Syntactic Analyzer, and Postprocessor [10]. The preprocessor extracts "primitives" from an input image and transforms the image to "language" sentences which are the inputs for the syntactic analyzer. This process usually involves a transformation operation, a threshold operation, and an averaging operation among others. The syntactic analyzer processes the input image by a set of grammatical rules inferred by a grammatical inference algorithm. This procedure is in order to accept those patterns which can be generated by the set of grammatical rules, and to reject all others. The postprocessor sequentially executes several information (semantic) rules. Those objects which are related, based on semantic information, are recognized from the image. Depending on the structures of the objects of interest, as indicated in Chapter 1, different types of grammatical rules are effective for describing and recognizing different objects. For example, any string-like object (such as a road or a river) is recognized more efficiently by a string grammar than a tree grammar. Whereas a tree grammar is necessary for recognizing a complex picture. The advantages of the string grammar approach lie in its systematic grammatical inference procedure and that the efficient syntactic analyzer, as finite state automata.

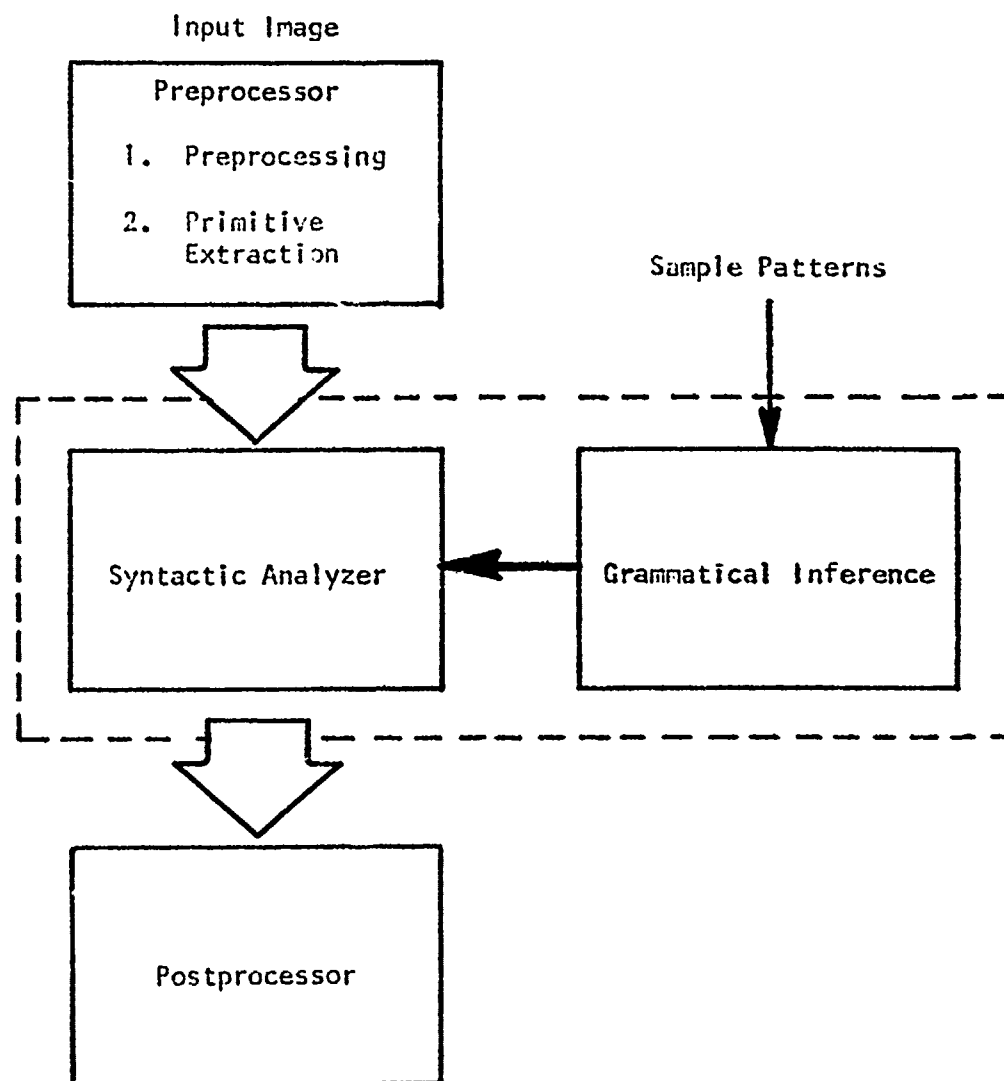


Figure 3.1. Flowchart for an image recognition system.

There are two major methods in the string grammar approach for image recognition; namely, the syntax-directed method and syntax-controlled method.

The syntax-directed method involves the following steps: first, an inference process is applied to a set of training imagery data to infer a set of grammatical rules which in turn formalizes a syntactic model. Second, based on this model, a set of template matching window patterns, which are generated by this grammar, is implemented to analyze the test images and to recognize the object of interest. Thus, the syntactic analyzer is a template matching process which is directed by the syntactic rules.

The syntax-controlled method is comprised of the following steps: first, a grammatical inference procedure is applied to a set of training imagery data to infer a set of grammatical rules which in turn is used to construct the recognizer or automaton. Then, this automaton is implemented to analyze the test images and to recognize the object of interest. Hence, the syntactic analyzer is an automaton which is directly controlled by the syntactic rules.

3.1. SYNTAX-DIRECTED METHOD FOR OBJECT RECOGNITION

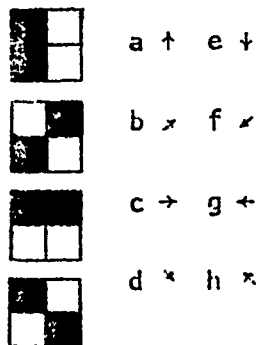
This research was motivated by the need for a method which can fully automate the recognition of such objects as highways, rivers, bridges, and commercial/industrial areas from satellite images such as those of LANDSAT. The statistical pattern recognition techniques which had been developed prior to this work had not shown satisfactory results. For example, the land-use classification of LANDSAT images has been studied by Todd and Baumgardner using spectral analysis [84]. It has been shown

that highways and other concrete areas, such as parking lots, could not be distinguished from each other due to the fact that both have similar spectral characteristics in spectral analysis. The utilization of the syntactic method to describe spatial relationship among different objects was suggested by Fu [3]. Some research was done on LANDSAT images by Brayer and Fu [14]. They were able to program a computer to analyze a city scene by constructing a hierarchical graph model which contains spatial distributions of all classes in the scene. Web grammars were used to describe spatial relationships between various objects in the scene. Li and Fu [15] started with pointwise statistical classification of LANDSAT images and then applied a tree system approach to the LANDSAT data interpretation. Bajcsy and Tavakoli [81] designed a computer program from the relational graph viewpoint to recognize objects from satellite pictures. The research undertaken here applies to the recognition of certain specific objects in LANDSAT images. The syntax-directed method utilizes a set of finite-state string grammar rules to describe and recognize the objects of interest.

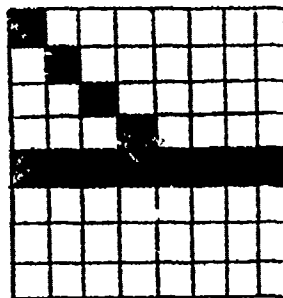
3.1.1. Grammatical Inference Process

Based on knowledge of highway structures, several initial grammatical rules were written. Then a training area was selected (which in this case was Lafayette, Indiana) and a preprocessed training image was obtained. The initial rules generated a set of pattern windows. The training image was then matched to the set of generated pattern windows to obtain the processed result. A highway map was used to evaluate the processed result. For the highway structures which existed in the map but not in the processed result, the grammatical rules to generate those

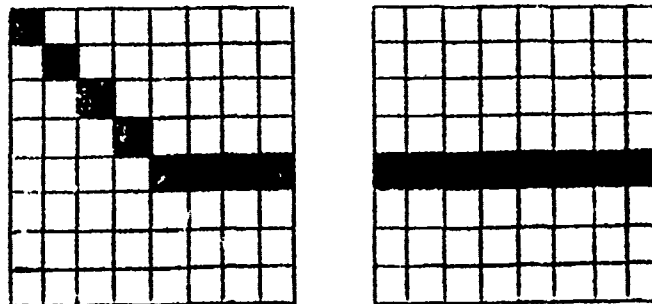
matching patterns were added to the initial set of rules and the image was then reprocessed. After several interactive steps the final set of grammatical rules was obtained. The primitives for the grammar are chosen as a, b, c, d, e, f, g, and h. These primitives are designed as 2 x 2 pixel blocks.



For example, a window for an intersection of highways might be



This syntactic analyzer was the window operation, which processes the image window by window. The movement of the window is to shift one column or one row at a time. Then multibranch patterns can be represented by one-branch grammar rules. For example, the window pattern mentioned above can be analyzed as the following two window patterns:



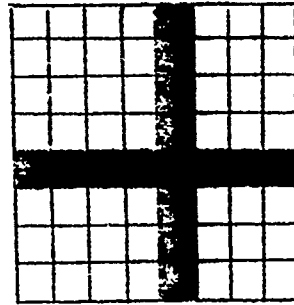
The one-branch grammar rules are as follows:

$$(1) \quad S \rightarrow dA_3 \quad A_3 \rightarrow dA_3 \quad A_3 \rightarrow dA_4 \quad A_4 \rightarrow cA_4 \quad A_4 \rightarrow c$$

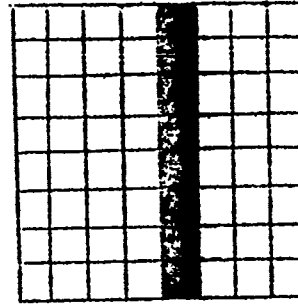
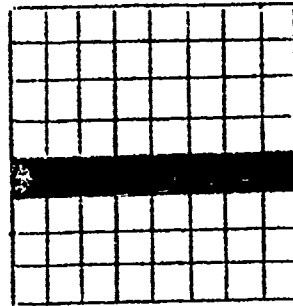
$$(2) \quad S \rightarrow cA_1 \quad A_1 \rightarrow cA_1 \quad A_1 \rightarrow cA_2 \quad A_2 \rightarrow cA_2 \quad A_2 \rightarrow c$$

These rules are finite-state grammar rules [2, 110].

A junction of two highways is as follows



can be analyzed by two window patterns that are generated by one-branch grammar rules. The two patterns are,



Then the resultant grammar rules can be expressed in terms of finite-state string grammar rules as follow. The grammar G is

$$G = (V_N, V_T, P, S)$$

$$V_N = \{S, A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10}\}$$

$$V_T = \{a, b, c, d, e, f, g, h\}$$

where P :

$S \rightarrow cA_1$	$A_1 \rightarrow cA_1$	$A_1 \rightarrow cA_2$	$A_2 \rightarrow cA_2$	$A_2 \rightarrow dA_2$	$A_2 \rightarrow eA_2$	$A_2 \rightarrow bA_2$
$S \rightarrow dA_3$	$A_3 \rightarrow dA_3$	$A_3 \rightarrow dA_4$	$A_4 \rightarrow dA_4$	$A_4 \rightarrow eA_4$	$A_4 \rightarrow fA_4$	$A_4 \rightarrow cA_4$
$S \rightarrow eA_5$	$A_5 \rightarrow eA_5$	$A_5 \rightarrow eA_6$	$A_6 \rightarrow eA_6$	$A_6 \rightarrow fA_6$	$A_6 \rightarrow gA_6$	$A_6 \rightarrow dA_6$
$S \rightarrow fA_7$	$A_7 \rightarrow fA_7$	$A_7 \rightarrow fA_8$	$A_8 \rightarrow fA_8$	$A_8 \rightarrow gA_8$	$A_8 \rightarrow hA_8$	$A_8 \rightarrow eA_8$

$S \rightarrow gA_9$ $A_9 \rightarrow gA_9$ $A_9 \rightarrow gA_{10}$ $A_{10} \rightarrow gA_{10}$ $A_{10} \rightarrow aA_{10}$ $A_{10} \rightarrow fA_{10}$ $A_{10} \rightarrow hA_{10}$
 $A_2 \rightarrow c$ $A_2 \rightarrow d$ $A_2 \rightarrow e$ $A_2 \rightarrow b$ $A_4 \rightarrow d$ $A_4 \rightarrow e$ $A_4 \rightarrow f$
 $A_4 \rightarrow c$ $A_6 \rightarrow e$ $A_6 \rightarrow f$ $A_6 \rightarrow g$ $A_6 \rightarrow d$ $A_6 \rightarrow cA_6$ $A_6 \rightarrow c$
 $A_8 \rightarrow f$ $A_8 \rightarrow g$ $A_8 \rightarrow h$ $A_8 \rightarrow e$ $A_{10} \rightarrow g$ $A_{10} \rightarrow a$ $A_{10} \rightarrow f$
 $A_{10} \rightarrow h$

3.1.2. Syntax-Directed Analysis

The syntax-directed method consists of two levels, namely, pre-processing and syntax-directed grammatical analysis. The transformation processor transforms the multispectral LANDSAT images into a single binary image. The syntax-directed analyzer then analyzes the transformed image based on a set of template matching patterns. Structures which are matched by this set of templates are accepted, otherwise they are rejected. The details of the preprocessing and syntax-directed grammatical analysis are given below:

(1) Preprocessing

A. Thresholding Process: First the LANDSAT images were defined in Euclidean n -dimensional space E^n . (The number n represents the number of channels to be chosen). A pixel is described by an ordered n -tuple (x_1, x_2, \dots, x_n) . LANDSAT measurements from channel 1 and 2 are very sensitive to concrete areas. A training area was chosen to establish the thresholds of the spectral intensity of concrete areas in channel 1 and 2. Then a threshold, H , was obtained from the sum of two thresholds from these channels. Since channels 3 and 4 are infrared bands [82], measurements from these two channels are sensitive to thermal emitting objects. Watery areas are strictly non-thermal emitting

objects. The negative reaction in the infrared bands causes contrast and makes the extraction of watery areas from channels 3 and 4 easier. The same procedure of threshold finding as that for concrete areas is applied here to obtain threshold, R , for river recognition. The transformation process works in such a way that if the sum of the spectral intensities of the pixels in the same position in two channels is greater than the sum of the two thresholds from the training area (for example, channels 1, 2, and threshold H for highway recognition, and channels 3, 4, and threshold R for river recognition), the pixel is set to 1; otherwise, it is set to 0. (For river recognition, the one-zero settings are reversed). Thus, the multispectral images are transformed to a single binary image.

It is true that both visible bands (channels 1 and 2 are sensitive to the concrete spectra. But in real world images, the influence of neighboring objects sometimes causes the deformation of the intensity of the object of interest (such as a highway). But when there is only one channel (image) available, the thresholding process can be designed by setting the threshold on that one image. Experiments of this kind were also conducted on other objects and it was shown that by using the sum of the spectral intensities of two visible channels (for highways) one obtains a more reliable result than that obtained by just setting a threshold on one channel. From the experiments of highway recognition on different LANDSAT images, it was shown that some highways are clear in channel 1 and some in channel 2. A weighted sum of threshold-finding could not be achieved but the results of summing the thresholds from the two channels were satisfactory. Hence, the sum of the thresholds from

the two channels was used as the threshold process in the experiments which will be shown here and in section 3.1.3.

B. Line Smoothing Process: After the thresholding process is completed, a line smoothing technique is applied to remove deformation and re-establish continuity of the lines. For a given center pixel of a 3×3 window, the operation starts from the left upper corner pixel. If it is one, the column is shifted. If it is zero, the surrounding eight pixels are checked. If there exists at least two "1's" which are not adjacent to each other, then a "1" is set into the center position. The operation continues until reaching the rightmost column of the digitized image. Then, the operation is shifted one row down and starts from the left most column with the same process until the last row of the digitized image is reached.

(2) Syntax-Directed Grammatical Analysis

Input: The transformed binary image which is a $Q(i,j)$ memory array.

Output: The syntax-directed result.

Algorithm:

Step 1: Set $G(M,N)$ to be an operation window (8×8 in size).

Step 2: Load that part of the array $Q(I,J)$ in which $J = 1, 8$; $I = 1, 8$ on the operation window $G(M,N)$.

Step 3: Compare the operation window with the set of template matching window patterns (Figure 3.2) which were generated by string grammar, G . (Thirty four templates were used in this analysis. The templates were obtained from the inference process. In the inference process, new templates are added to the set of templates in each interactive process. The number of templates used here (34) is the number of templates in the final interactive process of the inference process. From the experimental

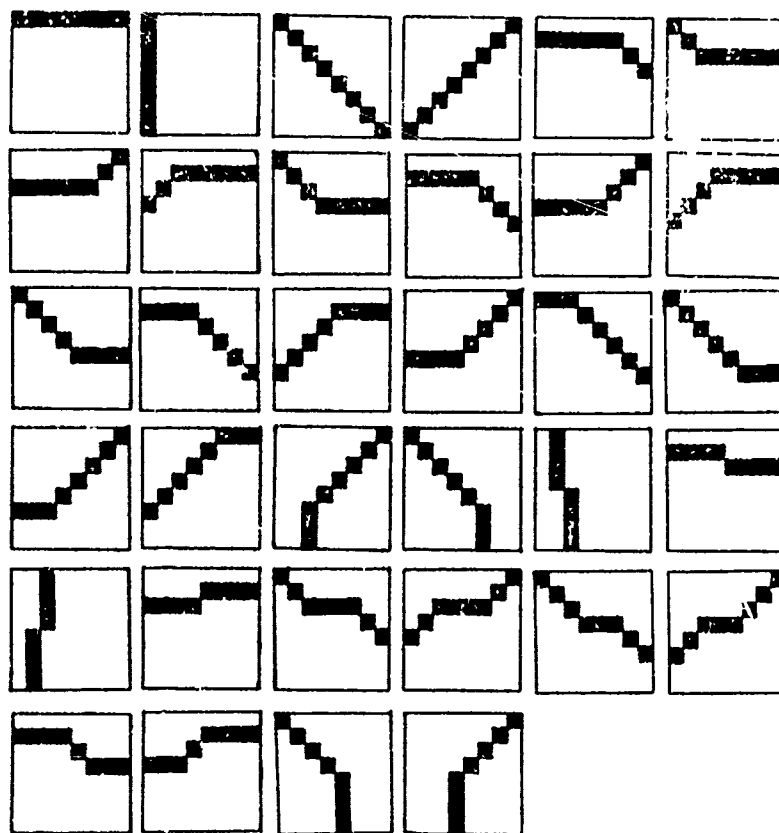


Figure 3.2. A set of template matching 8x8 window patterns.

result, these 34 templates proved adequate for these applications.) If the operation window belongs to any one of the pattern in that set of patterns, the primitive pattern in that window is accepted, and stored in the resulting memory array $R(I,J)$. If it does not belong to that set of patterns, then go to step 4.

Step 4: Shift one column to the right of $Q(I,J)$ in step 3. Then go to step 3 and continue until reaching the rightmost column.

Step 5: Shift one row downward; go to step 3, until reaching the last row of the digitized image. Syntactically correct structures are recognized and stored in the resultant memory array $R(I,J)$.

Step 6: Output the result, $R(I,J)$.

The flow chart for highway recognition by the syntax-directed method is given in Appendix B. Since rivers have the same linear features as highways, the inferred grammar for highways can also be used for rivers. This assumption is justified by the results of the experiments on river recognition. The flow chart for river recognition is also provided in Appendix C.

Spectrally speaking, bridges and commercial/industrial areas have similar characteristics to concrete parking lots and highways. This means that the existing statistical techniques are inadequate for the distinction of bridges from highways and for the recognition of commercial/industrial areas as such. The syntactic object recognition technique uses the spatial relationship to distinguish highways from other concrete areas by the syntax-directed method, and then uses semantic information as a postprocessor to distinguish bridges and commercial/industrial areas from the recognized highways. For bridge recognition,

first the images are processed by the syntax-directed method for highways and rivers. Then a logical "AND" operation is applied to the highway results "H" and the river result "R" to obtain the bridge recognition result "B". This operation results in a "bridge" which is the intersection of a "highway" and a "river". A flowchart of bridge recognition is given in Appendix D. The length of the recognized bridge can be calculated by the following algorithm:

Algorithm for calculation of bridge length and coordinates. (from LANDSAT data)

Input: Recognized bridge result.

Output: The calculated length and coordinates of the bridge.

Algorithm:

- (i) Calculate the number of horizontal rows which have at least one bridge pixel. The value is a.
- (ii) Calculate the number of vertical column which have at least one bridge pixel. The value is b.
- (iii) if $a = 1$, the length of the bridge, c, is $b \times 56$ meters.
If $b = 1$, the length of the bridge, c, is $a \times 79$ meters.
Otherwise go to (iv).
- (iv) The length of the bridge, c, $= \sqrt{(a \times 79)^2 + (b \times 56)^2}$.

The idea of this algorithm is to calculate the hypotenuse of a right triangle, using the information that each pixel in a LANDSAT image corresponding to 79 meters in vertical length and 56 meters in horizontal length. Step (iii) is the case in which a bridge is right on the horizontal row or vertical column. The coordinates of the recognized bridge can also be determined from the recognized bridge pixels, as the coordinates of the pixels in the image can be directly related to the global coordinates of meridian and elevation.

For commercial/Industrial area recognition, first the images are processed to obtain highways. The highway recognition result is then deleted from all concrete areas by the Boolean "Exclusive OR" operation. Then for the four by four cells of the image after the "Exclusive OR" operation, if over 60% of the pixels in the cell are concrete, this cell area is called commercial/industrial. This is because the commercial/industrial area has a high density of concrete buildings and concrete storage yard. The size of the recognized commercial/industrial area and the coordinates of the center of this area can be calculated by the following algorithm:

Algorithm for calculation of the area of a commercial/industrial area and its center. (from LANDSAT data)

Input: Commercial/industrial recognition result.

Output: The calculated area and the center of the commercial/industrial area.

Algorithm:

- (i) Calculate the number of commercial/industrial area pixels. The value is n .
- (ii) The area equals $n \times (0.079 \times 0.056)$ square kilometers.
- (iii) For every commercial/industrial area pixel $C_i(I, J)$ calculate $(\sum_{i=1}^n I_i)/n$. The value is P . Calculate $(\sum_{i=1}^n J_i)/n$. The value is q .
- (iv) The center of the commercial/industrial area is the coordinate (P, q) .

The resolution of LANDSAT images is 79 x 56 meters per pixel. Hence, the area of each pixel is (0.079×0.056) square kilometers. The center of commercial/industrial area is the coordinate which has the x coordinate

as the mean of all the x coordinates of the commercial/industrial pixels and the y coordinate as the mean of all the y coordinates of the commercial/industrial area pixels.

3.1.3. Computer Experimental Results on Recognition of Highway and River from Satellite Images

The syntax-directed method was implemented by FORTRAN programming on the IBM 360/67 computer at the Laboratory for Applications of Remote Sensing (LARS) at Purdue University.

(1) Highway Recognition

Figure 3.3(a) is a LANDSAT image of the Indianapolis, Indiana area. The platform altitude at which the picture was taken is 306,200 feet and there are 128 grey levels used in the digitized image [82]. Figure 3.3(b) is the intermediate output after line smoothing in the transformation process. The highway recognition result, by the syntax-directed method, is shown in Figure 3.3(c). This area is a 96 x 96 pixel image which shows the junction of Interstate highway 65 (northwest to southeast) and highway 465 (north to south) in the left upper part of Figure 3.3(d). The exact location of Figure 3.3(a) is labeled on Figure 3.3(d) as the square outlined. The experimental results indicate that the syntax-directed method is rather successful.

Figure 3.4(a) is a LANDSAT image of the downtown area of Chicago, Illinois. Figure 3.4(b) is the highway recognition result from Figure 3.4(a) by the syntax-directed method. This area is the junction of the Chicago Skyway and the Dan Ryan Expressway (highway 94) which is shown in the downtown map of Chicago Figure 3.4(c). Lake Shore Drive is also recognized, but as the resolution of the LANDSAT satellite sensor is only 79 x 56 meters per pixel, the recognition is not perfect,

because whenever the widths of highways are less than the resolution values, the recognition will be unfavorably affected. Figure 3.5(a) is the LANDSAT image of Harvey, Illinois. The computer result of highway recognition by the syntax-directed method is shown in Figure 3.5(b). For comparison of the accuracy of the result, the city map of Harvey, Illinois is shown in Figure 3.5(c). The recognized highways are highway 80 (southwest to northeast), highway 57 (south to north) and state road 50 (south to north). The "H" in Figures 3.3(c), 3.4(b), and 3.5(b) stands for a highway pixel.

(2) River Recognition

For the purpose of showing that this method works also for river recognition, a terrain area northeast of San Francisco, California was processed by the syntax-directed method for river recognition. The LANDSAT image is shown in Figure 3.6(a). The river recognition result given in Figure 3.6(b), shows that the syntax-directed method successfully recognized a winding river in that image. The size of this image is also 96 x 96 pixels. The topographic map for the area is shown in Figure 3.6(c).

Figure 3.7(a) is a LANDSAT image of the Lafayette, Indiana area. The river recognition result is shown in Figure 3.7(b). The "R" in Figure 3.7(b) stands for a river pixel. Figure 3.7(c) is a city map of Lafayette. The river recognition result shows the Wabash river through the Lafayette area. The Wabash river divides the area into West Lafayette and Lafayette. The lighter areas in the LANDSAT image are the concrete areas and the darker areas are the watery areas. The bright line from north to south in Figure 3.7(a) is highway 65, and the dark line from northeast to southwest is the Wabash River.



Figure 3.3(a). Satellite image of northwest part of Indianapolis, Indiana.

FIG. RESULT AFTER LINE SMOOTHING PROCESS

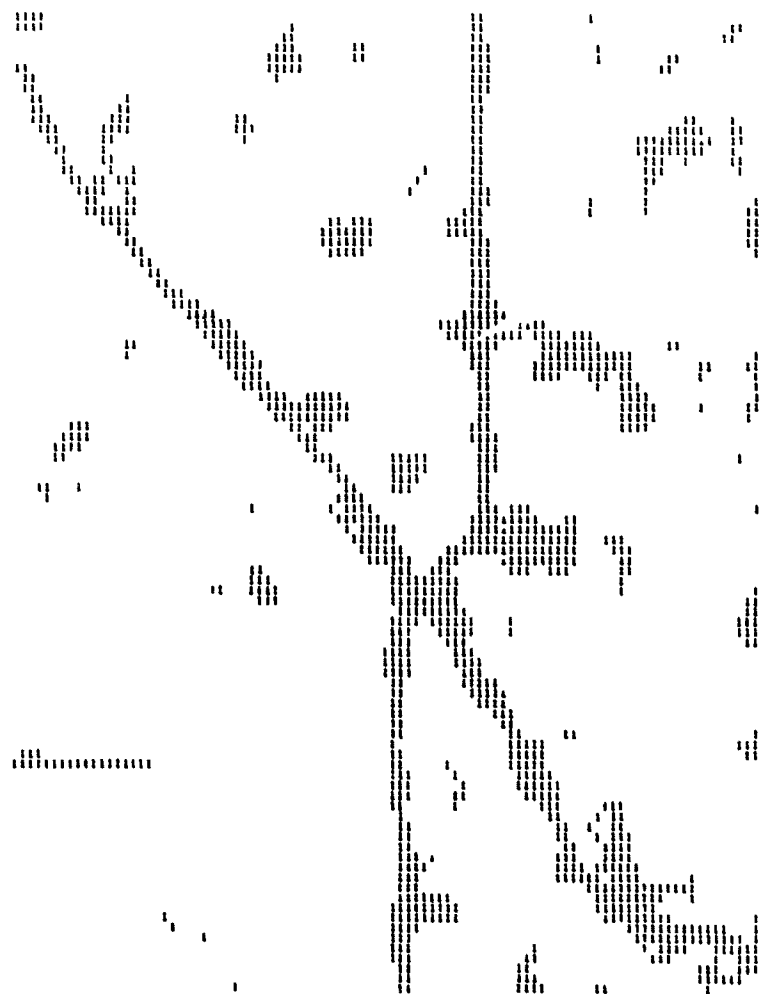


Figure 3.3(b). Intermediate output after line smoothing process on Figure 3.3(a).

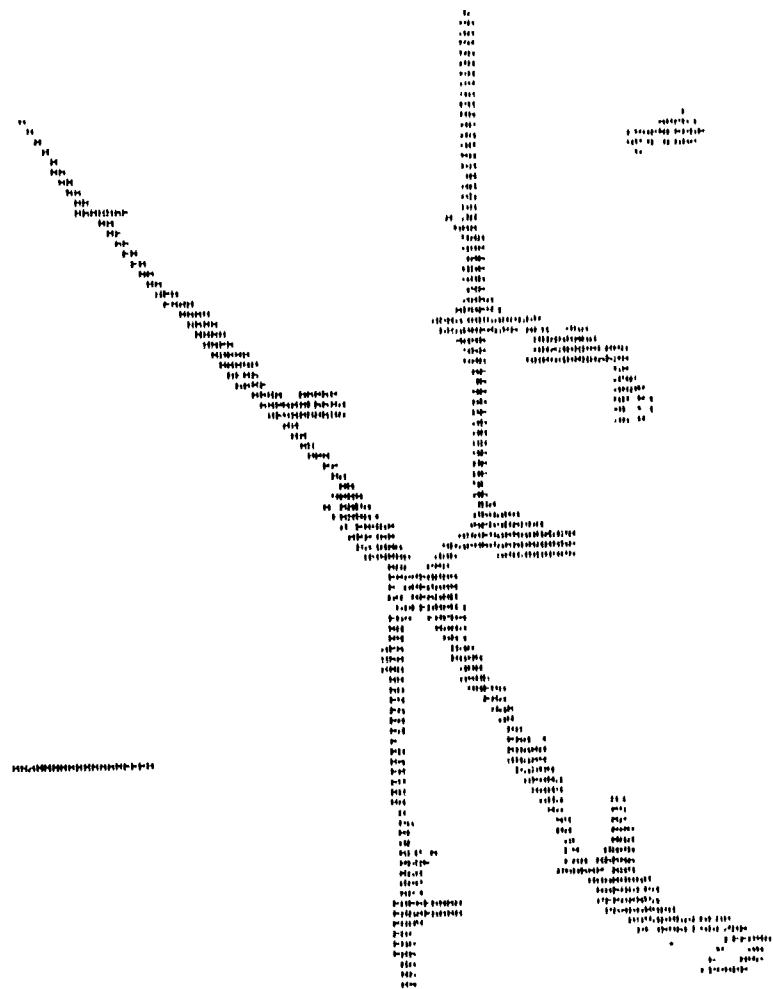


Figure 3.3(c). Highway recognition result from Figure 3.3(a) by syntax-directed method.

Fig. 3.3(a)

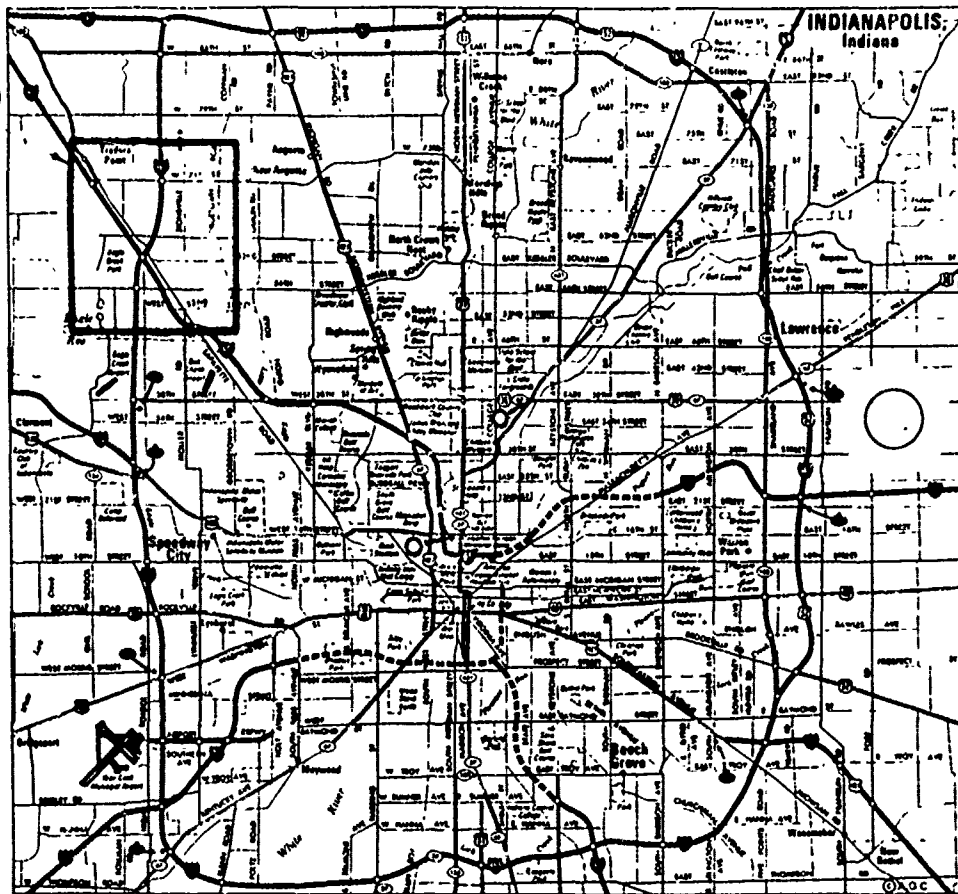


Figure 3.3(d). City map of Indianapolis, Indiana.

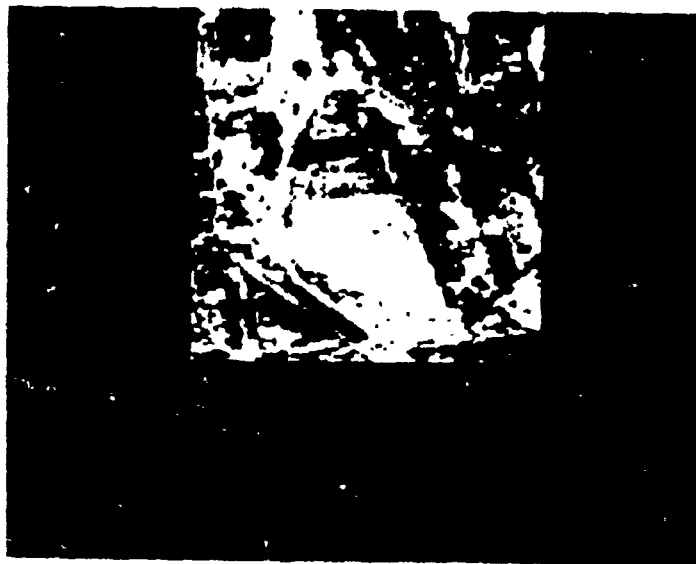


Figure 3.4(a). Satellite image of downtown Chicago, Illinois.



Figure 3.4(b). Highway recognition result from Figure 3.4(a) by syntax-directed method.

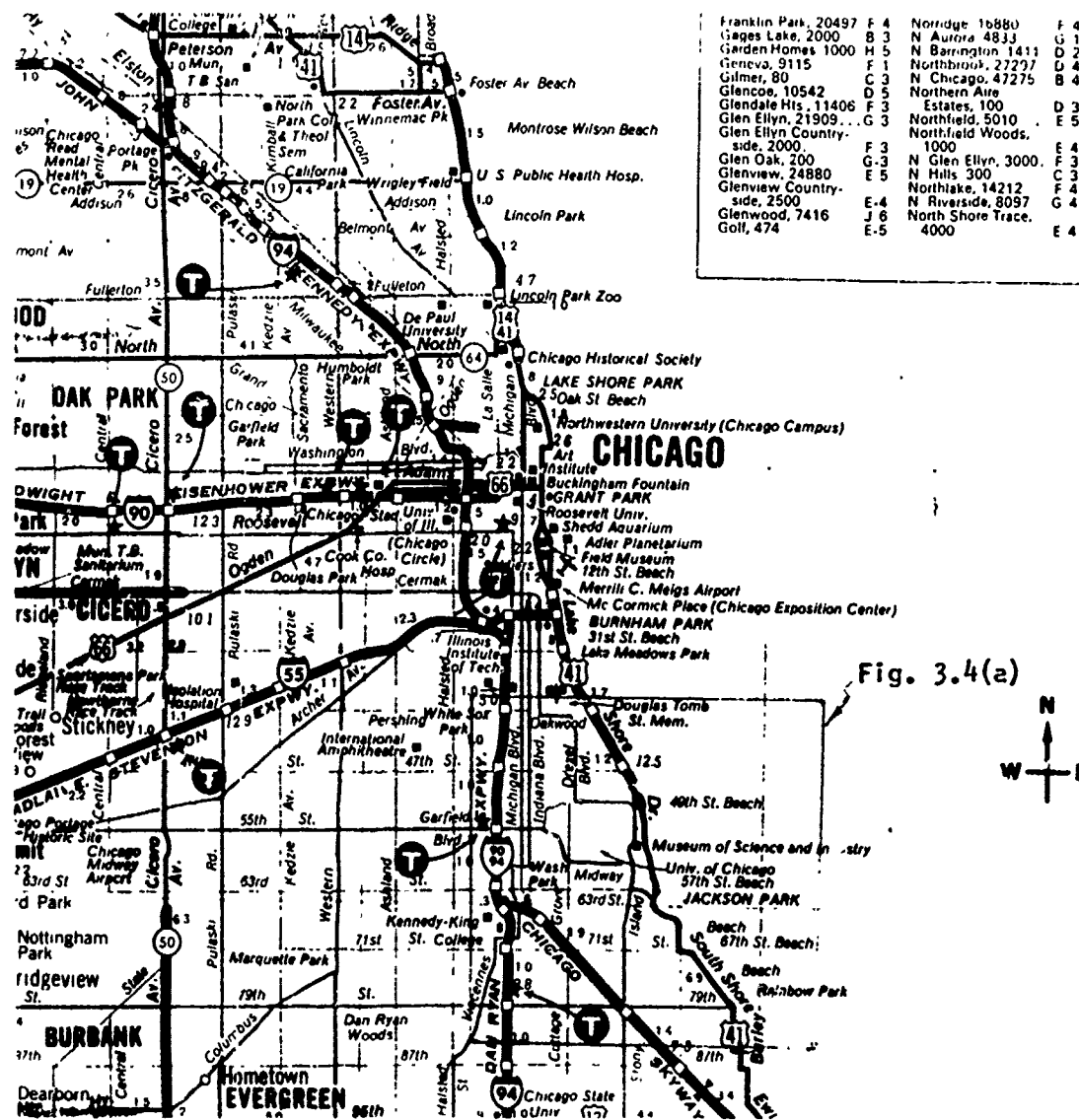


Figure 3.4(c). Map of downtown Chicago, Illinois.

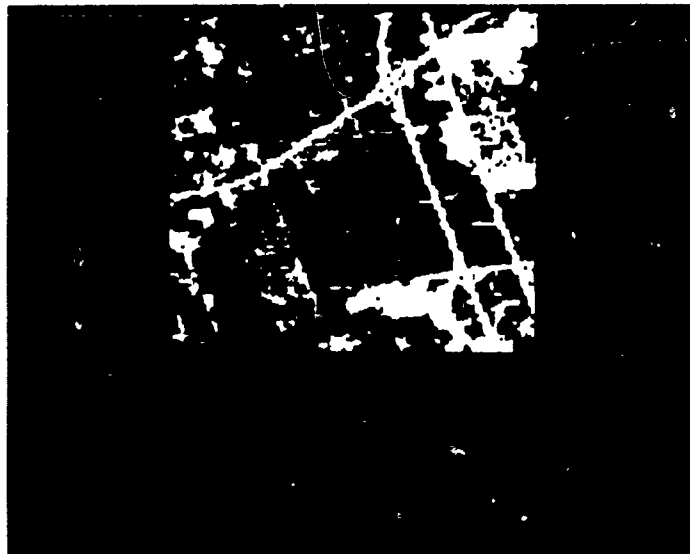


Figure 3.5(a). Satellite image of Harvey, Illinois.



Figure 3.5(b). Highway recognition result from Figure 3.5(a) by syntax-directed method.

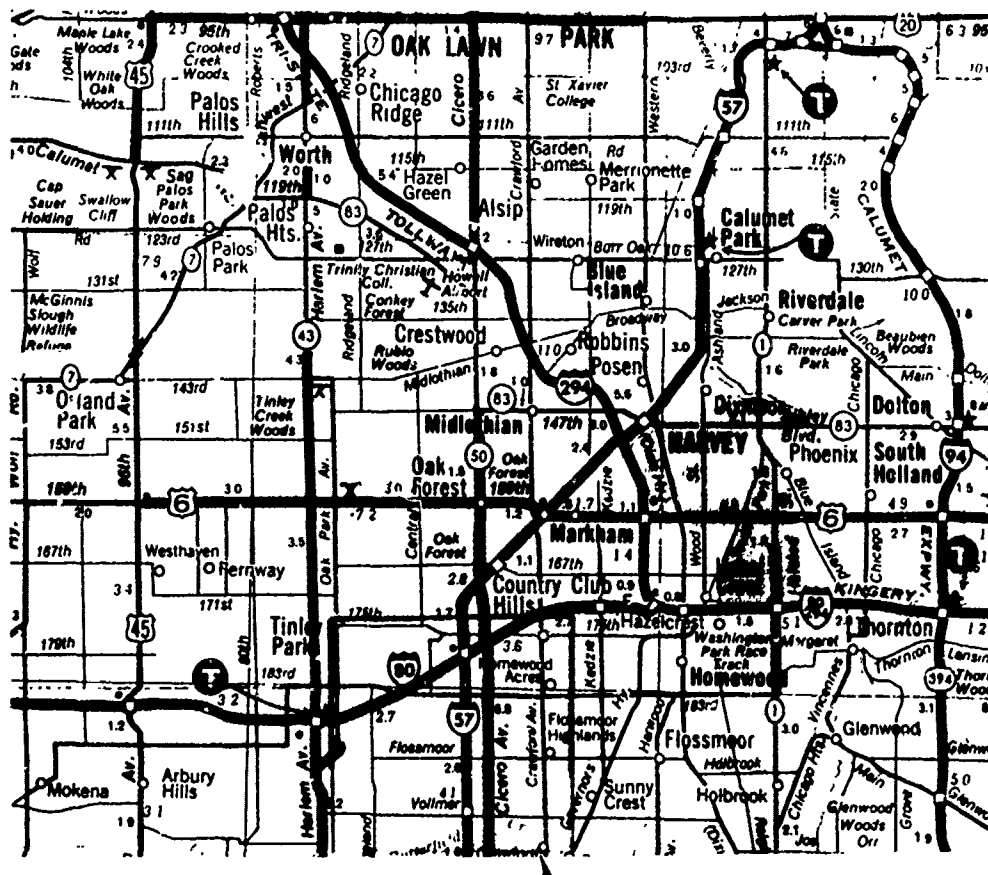


Fig. 3.5(a)

Figure 3.5(c). City map of Harvey, Illinois.



Figure 3.6(a). Satellite image of north part of San Francisco Bay area, California.



Figure 3.6(b). River recognition result from Figure 3.6(a) by syntax-directed method.

BEST AVAILABLE COPY

Fig. 3.6(a)

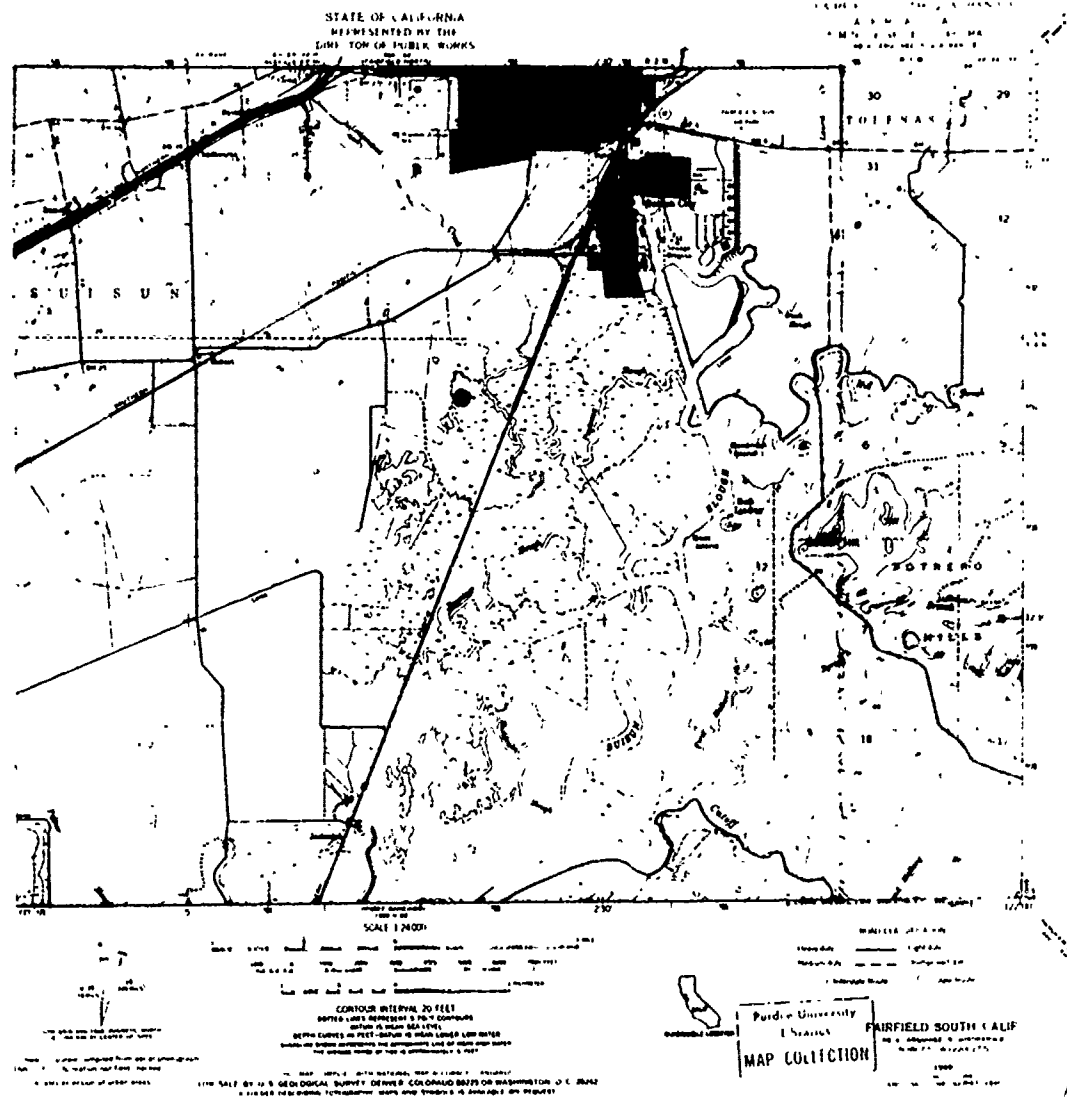


Figure 3.6(c). Topographic map of the same area of Figure 3.6(a).



Figure 3.7(a). Satellite image of Lafayette area, Indiana (192 x 192 pixels).

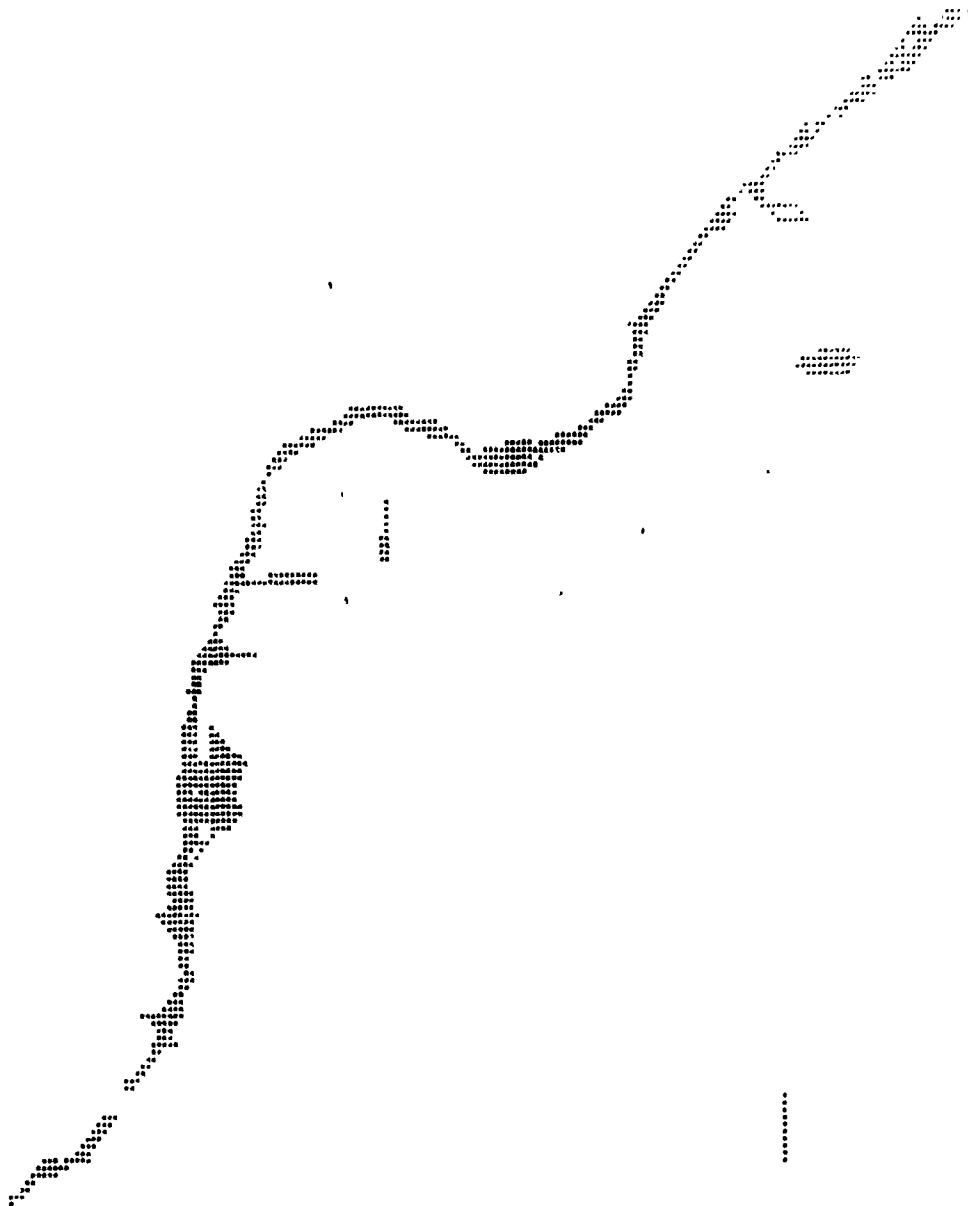


Figure 3.7(b). River recognition result from Figure 3.7(a) by syntax-directed method.

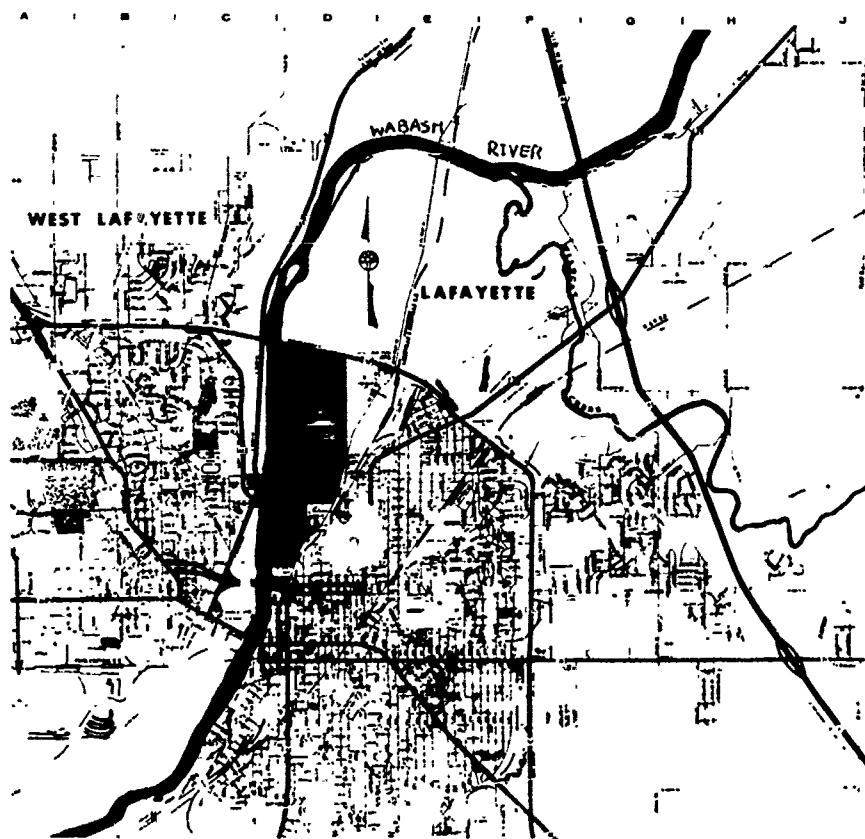


Figure 3.7(c). City map of Lafayette, Indiana.

3.1.4. Computer Experimental Results on Recognition of Bridges and Commercial/Industrial Area from Satellite Images

(1) Bridge Recognition

Figure 3.8(a) is a topographic map of the lower part of Figure 3.3 (a); Indianapolis, Indiana. It shows in the left part of the map, that there is a bridge over Eagle Creek Reservoir. The bridge recognition result by the syntactic object recognition method given in Figure 3.8(b) shows that the bridge was successfully recognized and that the length was calculated to be 672 meters. In the left lower part of Figure 3.8(a) the scale of the map is provided. The length shown in the map is about the same as that found by the syntactic method. The coordinates of the bridge can also be located by this method.

Another experiment was conducted in the Lafayette area pertaining to bridge recognition. Referring back to Figure 3.7(c), it can be seen that there is a bridge on Highway I-65 over the Wabash River. The LANDSAT image shown in Figure 3.9(a) processed by the syntax-directed method for bridge recognition and the result is shown in Figure 3.9(b). The recognized bridge is in the right lower part of the figure. Its length was calculated to be 454.1 meters. The coordinates of the location of the bridge are also given in Figure 3.9(b).

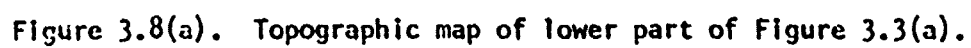
(2) Commercial/Industrial Area Recognition

Experiments were conducted in several different areas. Figure 3.10 (a) is a LANDSAT image (96 x 96 pixels) taken on September 30, 1972 of a section within the northwestern part of the Indianapolis area, which is a little south of the image in Figure 3.3(a). Figure 3.10(b) is the topographic map of the area. This map was made by the Department of Natural Resources of the state of Indiana in 1967. In Figure 3.10(c) the

intermediate output after preprocessing is given. Figure 3.10(d) is the highway recognition result. The intermediate output after the "Exclusive OR" operation in postprocessing is given in Figure 3.10(e). Figure 3.10(f) is the commercial/industrial area recognition result. This area is identified as the Lafayette Square Shopping Center in Indianapolis. Figure 3.10(g) is the urban development information extraction result. H, R, and C in Figure 3.10(g) represent highway, river, and commercial/industrial pixels, respectively. Interstate highway 65 (upper right part) goes into the city from the northwest. Highway 465 (from north to south) surrounds the city and highway 74 (left lower part) goes into the city from the west. The Lafayette Square Shopping Center is located in the suburb and close to the highways. The commercial/industrial area is automatically calculated to be 57.75 square kilometers. The center of the commercial/industrial area is calculated to be in coordinate (66,68) of the 96 x 96 image frame. This information was also automatically extracted by the method.

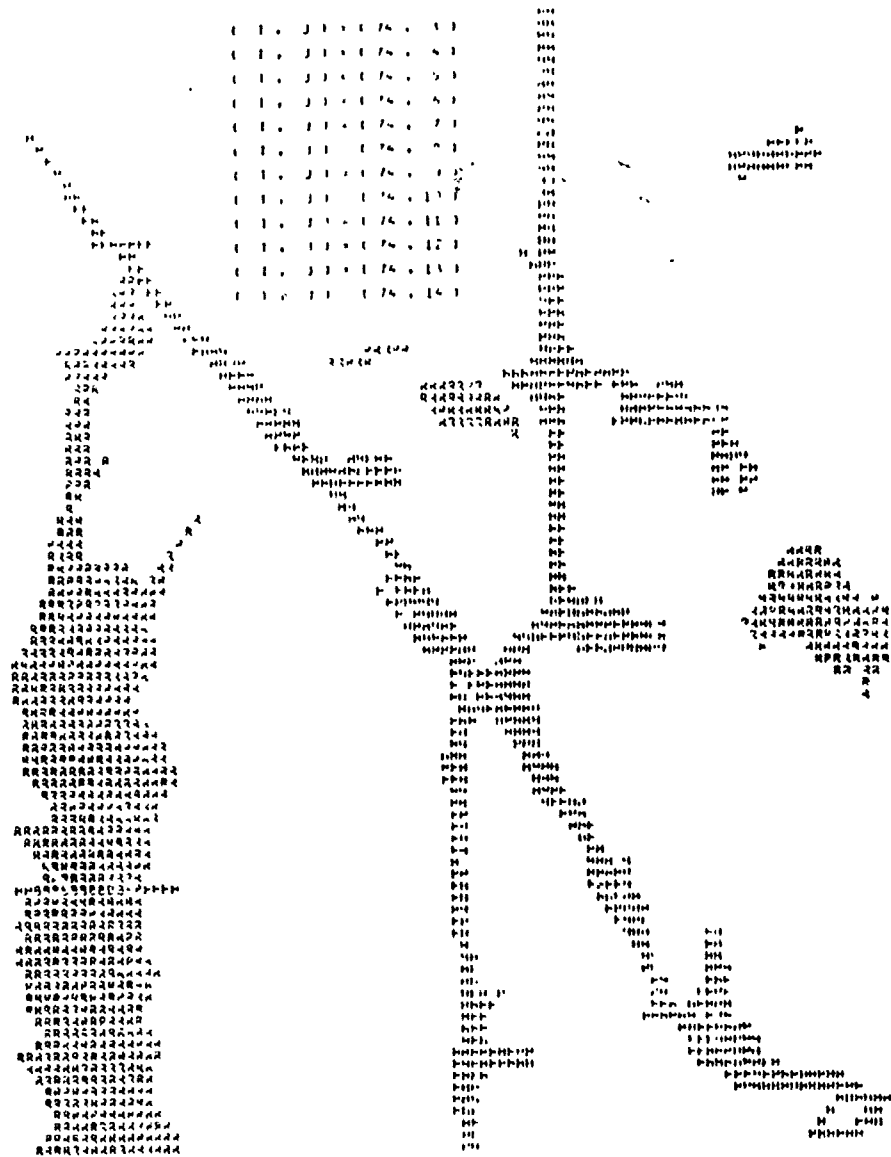
Comparing Figure 3.10(g) with the topographic map Figure 3.10(b) of 1967, we can observe the growth of the commercial area clearly on the northern and eastern parts of the shopping center. This indicates that the proposed method could be useful in topographic map making and updating.

Several large images (192 x 192 pixels) were also processed by the same technique. Figure 3.11(a) is the commercial/industrial area recognition result from the image of Lafayette, Indiana (compare Figure 3.7(a)). Figure 3.11(b) is the urban development information extraction result. The recognized commercial/industrial areas in Figure 3.11(a) are marked on the Lafayette map Figure 3.7(c). The commercial/industrial



BEST AVAILABLE COPY

THE CALCULATED LENGTH OF THE BRIDGE FROM THE SURVEYED POINTS



THE CALCULATED LENGTH OF THE BRIDGE FROM THE SURVEYED POINTS

Figure 3.8(b). Bridge recognition result from Figure 3.3(a).

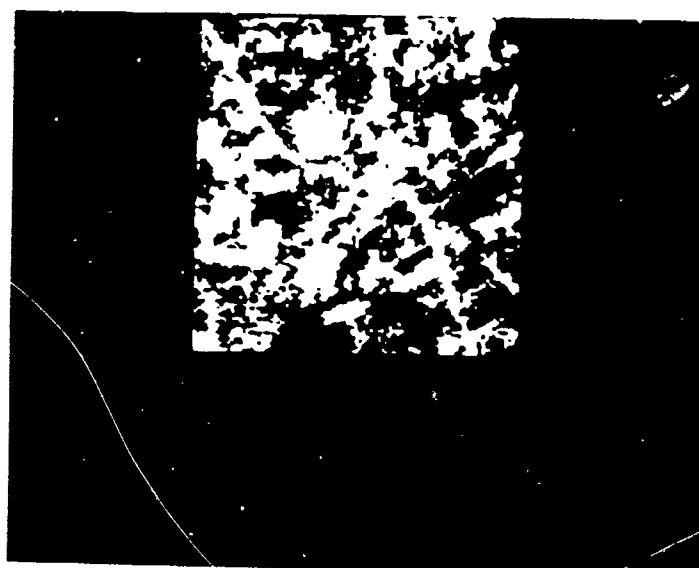


Figure 3.9(a). Satellite image of Lafayette area, Indiana.

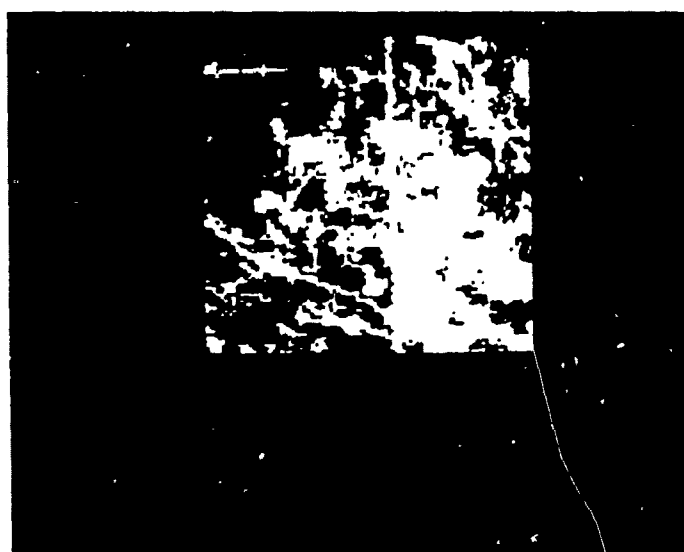


Figure 3.10(a). Satellite image of northwest part of Indianapolis, Indiana area (96 x 96 pixels).

STATE OF INDIANA
DEPARTMENT OF NATURAL RESOURCES
INDIANAPOLIS, INDIANA

CLERMONT QUADRANGLE
INDIANA
MINUTE 20E 3N 1E

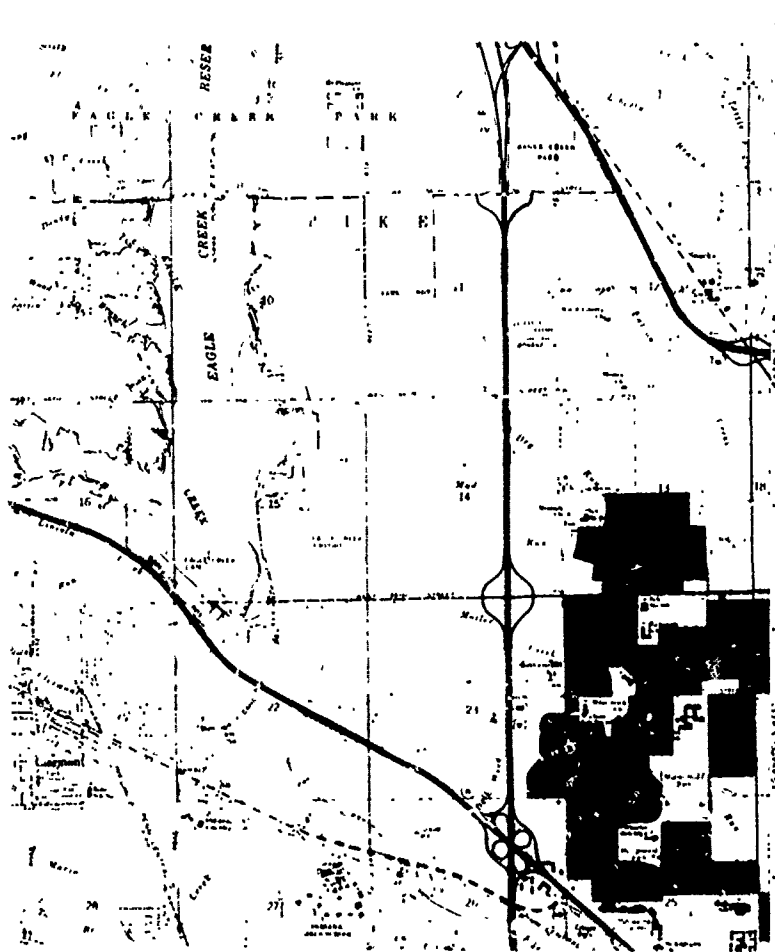


Figure 3.10(b). Topographic map of same area as Figure 3.10(a).



Figure 3.10(c). Intermediate output after preprocess of Figure 3.10(a).



Figure 3.10(d). Highway recognition result on Figure 3.10(a).



Figure 3.10(e). Intermediate output after exclusive or operation (EXOR) of the postprocessor.

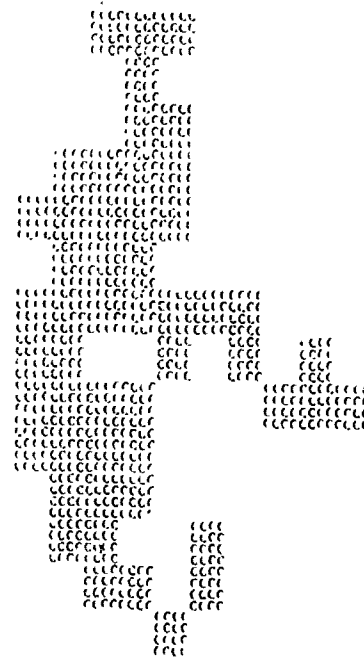


Figure 3.10(f). Commercial/industrial area recognition result.



Figure 3.10(g). Urban development information extraction result of Figure 3.10(a).

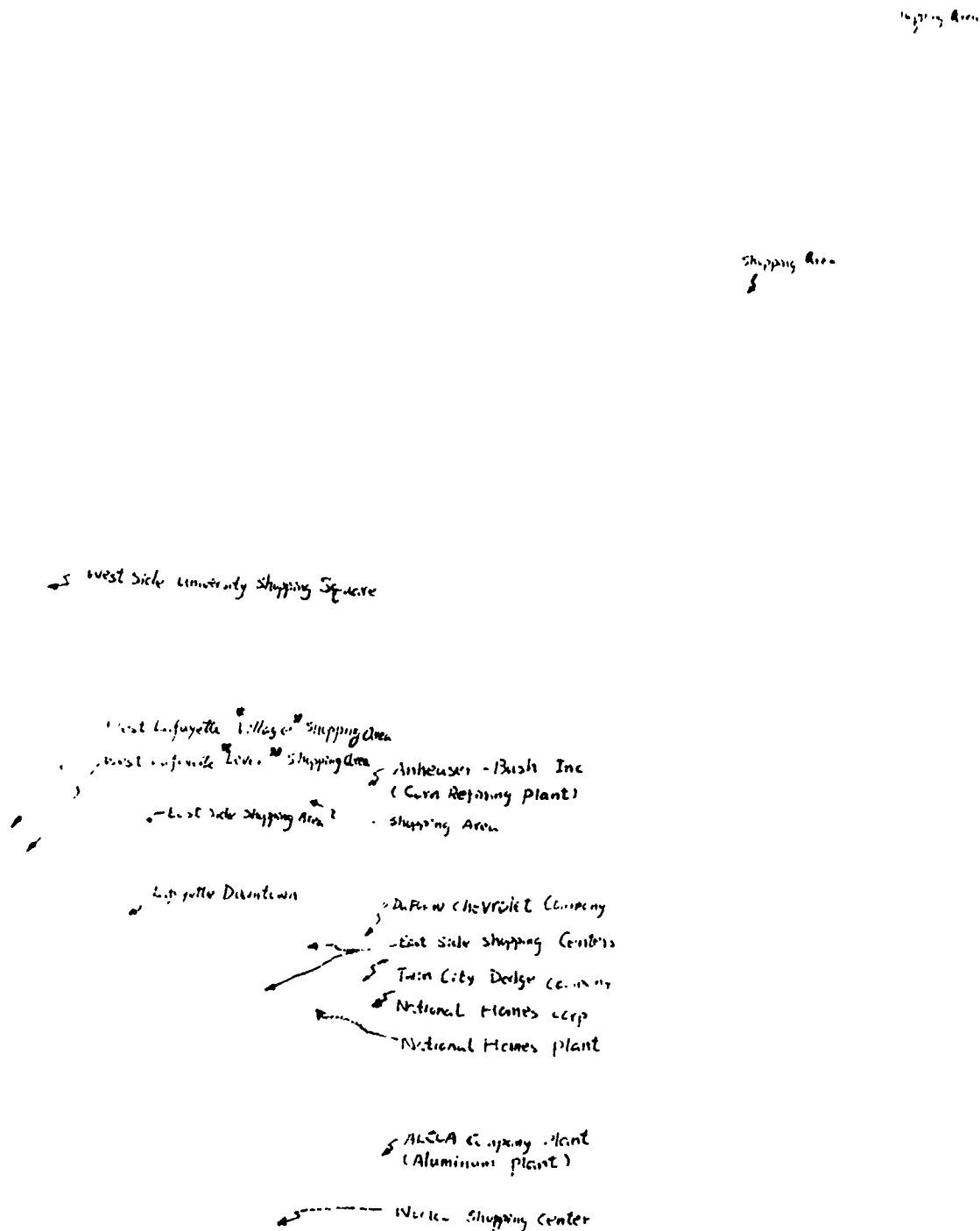


Figure 3.11(a). Commercial/Industrial area recognition result from Figure 3.7(a).



Figure 3.11(b). Urban development information extraction result of Figure 3.7(a).

areas of the Anheuser-Busch corn refining plant, the National Homes Company, the Alcoa Aluminum Plant, the West Lafayette Village and Levee shopping areas, the Woolco Shopping Center, and the Lafayette downtown area are successfully recognized.

3.2. SYNTAX-CONTROLLED METHOD FOR OBJECT RECOGNITION

The syntax-controlled method is a method which utilizes an automaton (or a parser) as a recognizer (or a syntax analyzer) to recognize the patterns of interest. The syntax-controlled method for image recognition differs from the syntax-directed method in the grammatical inference procedure and the syntactic analysis. The grammatical inference procedure for the syntax-controlled method is a fully computer automated finite-state grammar inference procedure whereas for the syntax-directed method, the grammatical inference procedure is an interactive process. The syntactic analyzer for the syntax-controlled method is a deterministic finite-state automaton which will be described in section 3.2.2., whereas for the syntax-directed method, the syntactic analysis is a template-matching process.

3.2.1. Grammatical Inference Procedure

In order to describe a class of patterns precisely, grammatical rules are inferred from a set of sample patterns. A k-tail method [20] which will be described in this section was implemented in this study. The grammatical rules which describe a class of patterns are automatically inferred from a set of training sample patterns by the computer program of the k-tail method. Before describing this technique, some definitions need to be given [20,110].

Definition 1. The formal derivative of a set of strings, A , with respect to the symbol $a \in V_T$ is defined as $DaA = \{x | ax \in A\}$.

Definition 2. If $a_1, a_2, \dots, a_{n-1} a_n$ is a string then, $Da_1 a_2, \dots, a_{n-1} a_n A = Da_n (Da_1, a_2, \dots, a_{n-1} A)$.

Definition 3. Let $u = a_1, a_2, \dots, a_r \in V_T^*$ and let $A \subseteq L(G)$. The k -tail of A with respect to u is defined as $(u, A, K) = \{x | x \in D_u A, |x| \leq K\}$ where $|x|$ is the length of the string.

Definition 4. Let u_i and u_j be two distinct states of the canonical derivative finite-state grammar G_{CD} . These two states are associated with the derivatives $Dx_i S^t$ and $Dx_j S^t$ respectively where x_i and x_j are sequences from V_T^* . The two states u_i and u_j are said to be k -tail equivalent if, and only if, $g(x_i, S^t, K) = g(x_j, S^t, K)$.

Definition 5. A non-deterministic finite-state automaton is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where

- (1) Q is a finite set of states;
- (2) Σ is a finite set of permissible input symbols;
- (3) δ is a mapping function from $Q \times \Sigma$ to $p(Q)$ which dictates the behavior of the finite-state control, (δ is sometimes called the state transition function);
- (4) q_0 in Q is the initial state of the finite-state control; and
- (5) $F \subseteq Q_0$ is the set of final states.

The inference procedure for the grammar which describes the highway and river structures is a k -tail method. The algorithm for this method is as follows:

Algorithm of k-tail grammatical inference procedure for finite-state string grammar.

Input: Training sample patterns.

Output: Finite-state string grammatical rules.

Algorithm:

- (i) Input training sample patterns and encode as the string of primitives a, b, c, d, e, f, g, and h.
- (ii) Find first order derivatives of the set of all strings A with respect to the primitive set a, b, c, d, e, f, g, and h. (e.g., $u_1 = D_a A = \{x | ax \in A\}$, $u_1 = D_a A$, $u_2 = D_b A$, $u_3 = D_c A$, $u_4 = D_d A$, $u_5 = D_e A$, $u_6 = D_f A$, $u_7 = D_g A$, and $u_8 = D_h A$).
- (iii) Find second order derivatives of the set of all strings A with respect to the primitive set. (e.g., $u_9 = D_{aa} A = D_a(D_a A) = \{x | aax \in A\}$.
 $u_9 = D_{aa} A$, $u_{10} = D_{ab} A$, $u_{11} = D_{ac} A$, ..., $u_{16} = D_{ah} A$,
 $u_{17} = D_{ba} A$, $u_{18} = D_{bb} A$, $u_{19} = D_{bc} A$, ..., $u_{24} = D_{bh} A$,
 $u_{25} = D_{ca} A$, $u_{26} = D_{cb} A$, $u_{27} = D_{cc} A$, ..., $u_{32} = D_{ch} A$,
 $u_{33} = D_{da} A$, $u_{34} = D_{db} A$, $u_{35} = D_{dc} A$, ..., $u_{40} = D_{dh} A$,
 $u_{41} = D_{ea} A$, $u_{42} = D_{eb} A$, $u_{43} = D_{ec} A$, ..., $u_{48} = D_{eh} A$,
 $u_{49} = D_{fa} A$, $u_{50} = D_{fb} A$, $u_{51} = D_{fc} A$, ..., $u_{56} = D_{fh} A$,
 $u_{57} = D_{ga} A$, $u_{58} = D_{gb} A$, $u_{59} = D_{gc} A$, ..., $u_{64} = D_{gh} A$,
 $u_{65} = D_{ha} A$, $u_{66} = D_{hb} A$, $u_{67} = D_{hc} A$, ..., $u_{72} = D_{hh} A$.
- (iv) Find third order derivatives of the set of all strings A, with respect to the primitive set a, b, c, d, e, f, g, and h. (e.g., $u_{74} = D_{aab} A = D_b(D_a(D_a A)) = \{x | aabx \in A\}$.
 $u_{73} = D_{aaa} A$, $u_{74} = D_{aab} A$, ..., $u_{80} = D_{aah} A$,

$$\begin{aligned}
 u_{81} &= D_{aba}A, u_{82} = D_{abb}A, \dots, u_{58} = D_{abh}A, \\
 &\cdot \\
 &\cdot \\
 &\cdot \\
 u_{568} &= d_{hga}A, u_{569} = D_{hgb}A, \dots, u_{575} = D_{hgh}A, \\
 u_{576} &= D_{hha}A, u_{577} = D_{hhb}A, \dots, u_{584} = D_{hhh}A.
 \end{aligned}$$

(v) Find n th order derivatives of the set of all strings A , with respect to the primitive set (N is the maximum length of string).

(vi) Find canonical derivative finite-state grammar.

$$G_{CD} = (V_T, V_N, R, \sigma)$$

V_T is the primitive set, $V_T = \{a, b, c, d, e, f, g, h\}$

$$V_N = \{u_1, u_2, u_3, \dots, u_{584}\}$$

$$\sigma = \{u_1, u_2, \dots, u_8\}$$

The rule, R , is defined as follows: let $u_i, u_j \in V_N$

$$u_i \rightarrow a u_j \text{ if, and only if, } D_a u_i = u_j$$

$$u_i \rightarrow a \text{ if, and only if, } \lambda \in D_a u_i$$

(e.g., $D_a u_i = u_9$ produces the grammatical rule $u_i \rightarrow a u_9$,

and $D_b u_{10} = u_{82}$ produces the grammatical rule $u_{10} \rightarrow b u_{82}$).

(vii) Find equivalence classes of states. Find the length of all the u_i where $i = 1$ to 584.

(viii) Find 1-tail equivalence classes. If $|u_i| > 1$ and $|u_j| > 1$, then $u_i = u_j$.

(ix) Find 2-tail equivalence classes. If $|u_i| > 2$ and $|u_j| > 2$, then $u_i = u_j$. If $|u_i| \leq 2$, then u_i has no equivalence classes.

(x) Find $(n-1)$ -tail equivalence classes. If $|u_i| > (N-1)$ and $|u_j| > (N-1)$, then $u_i = u_j$. If $|u_i| \leq (N-1)$ then u_i has no equivalence classes.

- (xi) Find the derived grammar, $G_D = (V_N, V_T, P, S)$, from the canonical derivative grammar, G_{CD} , and the minimized state set, u_i , by the equivalence classes.
- The terminal set, V_T , is the same for G_D and G_{CD} .
 - The non-terminal set, V_N , corresponds to the distinct blocks of a partition of V_{NC} by the equivalence classes.
 - S is the starting symbol and corresponds to the block which contains σ .
 - A grammatical rule of P is $B_i \rightarrow aB_j$ if, and only if, there exists $u_i, u_j \in V_N$ such that $u_i \rightarrow a u_j$, $u_i \in B_i$, $u_j \in B_j$.
 - A grammatical rule of P is $B_i \rightarrow a$ if, and only if, there exists $u_i \in V_N$ such that $u_i \rightarrow a$ and $u_i \in B_i$.
- (xii) Find the derived grammars, G_D , based on the partitions of 1-tail equivalence, 2-tail equivalence, and $(N-1)$ tail equivalence.
- (xiii) The obtained derived grammars, G_D , are the resultant finite-state string grammatical rules corresponding to the different values of k in the k -tail method.

This algorithm was implemented and applied to infer the highway grammar. Sample patterns are shown in Appendix E. The canonical derivative finite-state grammatical rules are inferred as the intermediate output of the finite-state string grammar. The canonical derivative grammatical rules are as follows.

$u_3 \rightarrow^c u_{27}$	$u_3 \rightarrow^c u_{27}$	$u_3 \rightarrow^c u_{27}$	$u_3 \rightarrow^c u_{28}$
$u_3 \rightarrow^c u_{27}$	$u_4 \rightarrow^c u_{36}$	$u_4 \rightarrow^c u_{35}$	$u_4 \rightarrow^d u_{36}$
$u_4 \rightarrow^d u_{36}$	$u_4 \rightarrow^c u_{35}$	$u_4 \rightarrow^d u_{36}$	$u_4 \rightarrow^d u_{36}$
$u_5 \rightarrow^e u_{45}$	$u_6 \rightarrow^f u_{54}$	$u_6 \rightarrow^g u_{55}$	$u_6 \rightarrow^f u_{54}$

$u_6 \rightarrow f \ u_{54}$	$u_6 \rightarrow g \ u_{55}$	$u_6 \rightarrow f \ u_{54}$	$u_6 \rightarrow f \ u_{54}$
$u_7 \rightarrow g \ u_{63}$	$u_7 \rightarrow g \ u_{63}$	$u_7 \rightarrow f \ u_{62}$	$u_7 \rightarrow g \ u_{63}$
$u_{27} \rightarrow c \ u_{219}$	$u_{219} \rightarrow c$	$u_{27} \rightarrow c \ u_{219}$	$u_{219} \rightarrow c$
$u_{27} \rightarrow d \ u_{220}$	$u_{220} \rightarrow d$	$u_{27} \rightarrow d \ u_{220}$	$u_{220} \rightarrow d$
$u_{23} \rightarrow d \ u_{228}$	$u_{228} \rightarrow d$	$u_{35} \rightarrow c \ u_{283}$	$u_{283} \rightarrow c$
$u_{36} \rightarrow d \ u_{292}$	$u_{292} \rightarrow d$	$u_{36} \rightarrow c \ u_{291}$	$u_{291} \rightarrow c$
$u_{36} \rightarrow d \ u_{292}$	$u_{292} \rightarrow d$	$u_{36} \rightarrow c \ u_{291}$	$u_{291} \rightarrow c$
$u_{36} \rightarrow e \ u_{293}$	$u_{293} \rightarrow e$	$u_{45} \rightarrow e \ u_{365}$	$u_{365} \rightarrow e$
$u_{44} \rightarrow f \ u_{438}$	$u_{438} \rightarrow f$	$u_{54} \rightarrow g \ u_{433}$	$u_{439} \rightarrow g$
$u_{54} \rightarrow f \ u_{438}$	$u_{438} \rightarrow f$	$u_{54} \rightarrow g \ u_{439}$	$u_{439} \rightarrow g$
$u_{54} \rightarrow e \ u_{437}$	$u_{437} \rightarrow e$	$u_{55} \rightarrow g \ u_{447}$	$u_{447} \rightarrow g$
$u_{55} \rightarrow g \ u_{447}$	$u_{447} \rightarrow g$	$u_{62} \rightarrow f \ u_{502}$	$u_{502} \rightarrow f$
$u_{63} \rightarrow g \ u_{511}$	$u_{511} \rightarrow g$	$u_{63} \rightarrow f \ u_{510}$	$u_{510} \rightarrow f$
$u_{63} \rightarrow f \ u_{510}$	$u_{510} \rightarrow f$	$u_{35} \rightarrow c \ u_{283}$	$u_{283} \rightarrow c$

After finding the equivalence classes of states u_i , the resultant finite-state string grammar for highways is inferred automatically by computer. The resultant grammar is as follows (when $k = 1$ in the k -tail method):

$$\begin{aligned}
 G_{k=i} &= (V_H, V_T, P, S) \\
 V_H &= \{S, A, B, C, D, E\} \\
 V_T &= \{a, b, c, d, e, f, g, h\} \\
 P: \quad &S \rightarrow cS \quad S \rightarrow dS \quad S \rightarrow eS \\
 &S \rightarrow fS \quad S \rightarrow gS \quad S \rightarrow cA
 \end{aligned}$$

$S \rightarrow dA$	$S \rightarrow dB$	$S \rightarrow eC$
$S \rightarrow fD$	$S \rightarrow gD$	$S \rightarrow fE$
$S \rightarrow gE$	$A \rightarrow c$	$A \rightarrow d$
$B \rightarrow d$	$C \rightarrow e$	$D \rightarrow f$
$D \rightarrow g$	$E \rightarrow f$	$E \rightarrow g$

The finite-state string grammar for highways when $k=2$ in the k -tail

method is also inferred as follows:

$G_{k=2} = (V_N, V_T, P, S)$		
$V_N = \{S, A, B, C, D, E, F, G, H, I, J, K, L, M, N\}$		
$V_T = \{a, b, c, d, e, f, g, h\}$		
$P:$	$S \rightarrow cA$	$S \rightarrow dB$
	$S \rightarrow cC$	$S \rightarrow eE$
	$S \rightarrow gG$	$S \rightarrow gI$
	$A \rightarrow cJ$	$A \rightarrow dJ$
	$C \rightarrow cJ$	$D \rightarrow dJ$
	$D \rightarrow eL$	$E \rightarrow eL$
	$F \rightarrow gM$	$F \rightarrow eL$
	$H \rightarrow fN$	$I \rightarrow gN$
	$J \rightarrow c$	$J \rightarrow d$
	$L \rightarrow e$	$M \rightarrow f$
	$N \rightarrow f$	$N \rightarrow g$

An excellent aspect of the k -tail method lies in the fact that the value of k can vary depending on the case. The grammar inferred by $k=2$ is more precise than the grammar inferred by $k=1$ in describing the highway structures. However, the computer processing time of grammatical inference for $k=2$ is longer than the time of grammatical inference for $k=1$. The CPU time for grammatical inference of $k=1$ is 9 seconds. The CPU time for grammatical inference of $k=2$ is 15 seconds on the IBM 360/

67 computer. The grammatical inference procedure of $k=3$ was also implemented. The CPU time is 270% of that for $k=1$. Thus, the case of $k=3$ is not suggested for highway grammar inference. Concerning the choice between $k=1$ and $k=2$ for the highway grammar inference, several computer experiments of highway recognition by these two sets of grammars were conducted. The experiments and the suggested choice of k values will be presented in the next section.

3.2.2. Syntax-Controlled Analysis

The syntax-controlled method consists of two levels, namely, a transformation process and the syntax-controlled analysis. The transformation process of the syntax-controlled method is the same as that for the syntax-directed method given in section 3.1.2. The syntax-controlled analysis uses a deterministic finite-state automaton as a recognizer to analyze the transformed image. Instead of comparing the operation window $G(M,N)$ with the set of template windows as in section 3.1.2, the window $G(M,N)$ is analyzed by the finite-state automaton. The algorithm is as follows:

Algorithm of highway recognition by syntax-controlled method.

Input: LANDSAT images.

Output: Highway recognition result by syntax-controlled method.

Algorithm:

- (i) Transformation Process: same as for syntax-directed method as in section 3.1.2.
- (ii) Finite-state Automaton Analysis: computer automatically constructs the finite-state automaton from the inferred grammar.
- (iii) Construct the finite-state automaton $M = (\Sigma, Q, \delta, q_0, F)$ from the finite state grammar $G = (V_N, V_T, P, S)$ as follows [2]:

- a. $\Sigma = V_T$
 - b. $Q = V_N \cup \{T\}$
 - c. $q_0 = s$
 - d. If P contains the production $S \rightarrow \lambda$, then $F = \{S, T\}$.
Otherwise $F = \{T\}$.
 - e. If $B \rightarrow a$, $B \in V_N$, $a \in V_T$ is in P of grammar G , then
 $\delta(B, a) = T$.
 - f. If $B \rightarrow aC$ is in P of grammar G and $C \in V_N$, $a \in V_T$, then
 $\delta(B, a) = C$ and $\delta(T, a) = \emptyset$ for each $a \in V_T$.
- (iv) Construct the deterministic finite automaton M' from the finite state automaton M as follows [2]:
- $$M' = (\Sigma', Q', \delta', q_0', F') \quad (M = (\Sigma, Q, \delta, q_0, F))$$
- a. The states (elements of Q') of M' are all the subsets of Q .
 - b. $\Sigma' = \Sigma$
 - c. F' is the set of all states in Q' containing a state of F .
 - d. $q_0' = [q_0]$, a state of M' is denoted by $[q_1, q_2, \dots, q_i] \in Q'$ where $q_1, q_2, \dots, q_i \in Q$.
 - e. $\delta'([q_1, \dots, q_i], a) = [p_1, p_2, \dots, p_j]$ if, and only if,
 $\delta([q_1, \dots, q_i], a) = \bigcup_{k=1}^i \delta(q_k, a) = \{p_1, p_2, \dots, p_j\}$.
- (v) Set $G(M, N)$ to be an operation window (8×8) and load the array of the transformed image from Step (i). $G(M, N)$ equals $Q(I, J)$ where $J = 1, 8$; and $I = 1, 8$ for the first case.
- (vi) Encode the patterns of the $G(M, N)$ window into the string of the primitives $a\downarrow$, $b\downarrow$, $c\downarrow$, $d\downarrow$, $e\downarrow$, $f\downarrow$, $g\downarrow$, and $h\downarrow$.
- (vii) Send the string to the deterministic finite-state automaton M' . If the automaton accepts the string, then this pattern is accepted. Therefore, proceed to the next step, otherwise this pattern is rejected. Proceed to the next step.

- (viii) Shift one column to the right of $Q(l,J)$ in step (iii). Then go to step (v) and continue until reaching the rightmost column.
- (ix) Shift one row downward in step (v); go to step (v) until reaching the last row of the digitized image. Syntactically correct structures are recognized and stored in the resultant memory array $R(l,J)$.
- (x) Output the result, $R(l,J)$, which is the result of highway recognition by the syntax-controlled method.

For the highway grammar ($k=1$), the finite-state automaton is automatically constructed by computer as follows:

Finite-state automaton for highway recognition $M_1 = (\Sigma, Q, \delta, q_0, F)$

$$\Sigma = \{a, b, c, d, e, f, g, h\}$$

$$Q = V_N \cup \{T\}$$

$$q_0 = S$$

$$F = \{T\}$$

$$\delta: \delta(S,c) = S$$

$$\delta(S,d) = S$$

$$\delta(S,e) = S$$

$$\delta(S,f) = S$$

$$\delta(S,g) = S$$

$$\delta(S,c) = A$$

$$\delta(S,d) = A$$

$$\delta(S,d) = B$$

$$\delta(S,e) = C$$

$$\delta(S,f) = D$$

$$\delta(S,g) = D$$

$$\delta(S,f) = E$$

$$\delta(S, g) = E$$

$$\delta(A, c) = T$$

$$\delta(B, d) = T$$

$$\delta(A, d) = T$$

$$\delta(C, e) = T$$

$$\delta(D, f) = T$$

$$\delta(D, g) = T$$

$$\delta(E, f) = T$$

$$\delta(E, g) = T$$

$$\delta(T, a) = \phi$$

$$\delta(T, b) = \phi$$

$$\delta(T, c) = \phi$$

$$\delta(T, d) = \phi$$

$$\delta(T, e) = \phi$$

$$\delta(T, f) = \phi$$

$$\delta(T, g) = \phi$$

$$\delta(T, h) = \phi$$

The corresponding deterministic finite-state automaton for the case of the grammar, $k=1$, is as follows: $M_1' = (\Sigma', Q', \delta', q_0', F')$

$$(M_1 = (\Sigma, Q, \delta, q_0, F)).$$

$$\Sigma' = \Sigma = \{a, b, c, d, e, f, g, h\}$$

$$Q' = V_H \cup \{T, \phi\} \cup \{[S, A], [S, A, B], [S, C], [S, D, E], [S, A, T], [S, A, B, T], [S, C, T], [S, D, E, T]\}$$

$$q_0' = [S]$$

$$F' = \{T, [S, A, T], [S, A, B, T], [S, C, T], [S, D, E, T]\}$$

$$\delta' : \delta([S], c) = [S, A]$$

$$\delta([S, c], g) = [S, D, E]$$

$$\delta([S], d) = [S, A, B]$$

$$\delta([S, D, E], c) = [S, A]$$

$\delta([S], e) = [S, C]$	$\delta([S, D, E], d) = [S, A, B]$
$\delta([S], f) = [S, D, E]$	$\delta([S, D, E], e) = [S, C]$
$\delta([S], g) = [S, D, E]$	$\delta([S, D, E], f) = [S, D, E, T]$
$\delta([A], c) = [T]$	$\delta([S, D, E], g) = [S, D, E, T]$
$\delta([A], d) = [T]$	$\delta([S, A, T], c) = [S, A, T]$
$\delta([B], d) = [T]$	$\delta([S, A, T], d) = [S, A, B]$
$\delta([C], e) = [T]$	$\delta([S, A, T], e) = [S, C]$
$\delta([D], f) = [T]$	$\delta([S, A, T], f) = [S, D, E]$
$\delta([D], g) = [T]$	$\delta([S, A, T], g) = [S, D, E]$
$\delta([E], f) = [T]$	$\delta([S, A, B, T], c) = [S, A, T]$
$\delta([E], g) = [T]$	$\delta([S, A, B, T], d) = [S, A, B, T]$
$\delta([T], c) = [\phi]$	$\delta([S, A, B, T], e) = [S, C]$
$\delta([T], d) = [\phi]$	$\delta([S, A, B, T], f) = [S, D, E]$
$\delta([T], e) = [\phi]$	$\delta([S, A, B, T], g) = [S, D, E]$
$\delta([T], f) = [\phi]$	$\delta([S, C, T], c) = [S, A]$
$\delta([T], g) = [\phi]$	$\delta([S, C, T], d) = [S, A, B]$
$\delta([S, A], c) = [S, A, T]$	$\delta([S, C, T], e) = [S, C, T]$
$\delta([S, A], d) = [S, A, B, T]$	$\delta([S, C, T], f) = [S, D, E]$
$\delta([S, A], e) = [S, C]$	$\delta([S, C, T], g) = [S, D, E]$
$\delta([S, A], f) = [S, D, E]$	$\delta([S, D, E, T], c) = [S, A]$
$\delta([S, A], g) = [S, D, E]$	$\delta([S, D, E, T], d) = [S, A, B]$
$\delta([S, A, B], c) = [S, A, T]$	$\delta([S, D, E, T], e) = [S, C]$
$\delta([S, A, B], d) = [S, A, B, T]$	$\delta([S, D, E, T], f) = [S, D, E, T]$
$\delta([S, A, B], e) = [S, C]$	$\delta([S, D, E, T], g) = [S, D, E, T]$
$\delta([S, A, B], f) = [S, D, E]$	$\delta([T], a) = [\phi]$
$\delta([S, A, B], g) = [S, D, E]$	$\delta([T], b) = [\phi]$

$$\delta([S,C],c) = [S,A]$$

$$\delta([T],h) = [\phi]$$

$$\delta([S,C],d) = [S,A,B]$$

$$\delta([S,C],f) = [S,D,E]$$

$$\delta([S,C],e) = [S,C,T]$$

The finite-state automaton for highway recognition based on the inferred grammar ($k=2$) is as follows: $M_2 = (\Sigma, Q, \delta, q_0, F)$.

$$\Sigma = \{a, b, c, d, e, f, g, h\}$$

$$Q = V_N \cup \{T\}$$

$$q_0 = S$$

$$F = \{T\}$$

$$\delta = \delta(S,c) = A$$

$$\delta(S,d) = B$$

$$\delta(S,d) = D$$

$$\delta(S,c) = C$$

$$\delta(S,e) = E$$

$$\delta(S,f) = F$$

$$\delta(S,g) = G$$

$$\delta(S,g) = J$$

$$\delta(S,f) = H$$

$$\delta(A,c) = J$$

$$\delta(A,d) = J$$

$$\delta(B,d) = K$$

$$\delta(C,c) = J$$

$$\delta(D,d) = J$$

$$\delta(D,c) = J$$

$$\delta(D,e) = L$$

$$\delta(E,e) = L$$

$$\delta(F, f) = M$$

$$\delta(F, g) = M$$

$$\delta(F, e) = L$$

$$\delta(G, g) = M$$

$$\delta(H, f) = N$$

$$\delta(I, g) = N$$

$$\delta(I, f) = M$$

$$\delta(J, c) = T$$

$$\delta(J, d) = T$$

$$\delta(K, d) = T$$

$$\delta(L, e) = T$$

$$\delta(M, f) = T$$

$$\delta(M, g) = T$$

$$\delta(N, f) = T$$

$$\delta(N, g) = T$$

$$\delta(T, a) = \phi$$

$$\delta(T, b) = \phi$$

$$\delta(T, c) = \phi$$

$$\delta(T, d) = \phi$$

$$\delta(T, e) = \phi$$

$$\delta(T, f) = \phi$$

$$\delta(T, g) = \phi$$

$$\delta(T, h) = \phi$$

The corresponding deterministic finite-state automaton, M' , of the above finite-state automaton from the inferred grammar, ($k=2$), is as follows: $M_2' = (\Sigma', Q', \delta', q_0', F')$ ($M_2 = (\Sigma, Q, \delta, q_0, F)$).

$$\Sigma' = \Sigma = \{a, b, c, d, e, f, g, h\}$$

$$Q' = V_N \cup \{T, \phi\} \cup \{[A, C], [B, D], [F, H], [G, I], [K, J], [M, N]\}$$

$$q_0' = [S]$$

$$F' = [T]$$

$$\delta' : \delta([S], c) = [A, C]$$

$$\delta([S], d) = [B, D]$$

$$\delta([S], e) = [E]$$

$$\delta([S], f) = [F, H]$$

$$\delta([S], g) = [G, I]$$

$$\delta([A], c) = [J]$$

$$\delta([A], d) = [J]$$

$$\delta([B], d) = [K]$$

$$\delta([C], c) = [J]$$

$$\delta([D], c) = [J]$$

$$\delta([D], d) = [J]$$

$$\delta([D], e) = [L]$$

$$\delta([E], e) = [L]$$

$$\delta([F], e) = [L]$$

$$\delta([F], f) = [H]$$

$$\delta([F], g) = [M]$$

$$\delta([G], g) = [M]$$

$$\delta([H], f) = [N]$$

$$\delta([I], f) = [M]$$

$$\delta([I], g) = [N]$$

$$\delta([J], c) = [T]$$

$$\delta([J], d) = [T]$$

$$\delta([M], g) = [T]$$

$$\delta([N], f) = [T]$$

$$\delta([N], g) = [T]$$

$$\delta([T], a) = [\phi]$$

$$\delta([T], b) = [\phi]$$

$$\delta([T], c) = [\phi]$$

$$\delta([T], d) = [\phi]$$

$$\delta([T], e) = [\phi]$$

$$\delta([T], f) = [\phi]$$

$$\delta([T], g) = [\phi]$$

$$\delta([A, C], c) = [J]$$

$$\delta([A, C], d) = [J]$$

$$\delta([B, D], c) = [J]$$

$$\delta([B, D], d) = [K, J]$$

$$\delta([B, D], e) = [L]$$

$$\delta([F, H], e) = [L]$$

$$\delta([F, H], f) = [M, N]$$

$$\delta([F, H], g) = [M]$$

$$\delta([G, I], f) = [M]$$

$$\delta([G, I], g) = [M, N]$$

$$\delta([K, J], c) = [T]$$

$$\delta([K, J], d) = [T]$$

$$\delta([K],d) = [T]$$

$$\delta([M,N],f) = [T]$$

$$\delta([L],e) = [T]$$

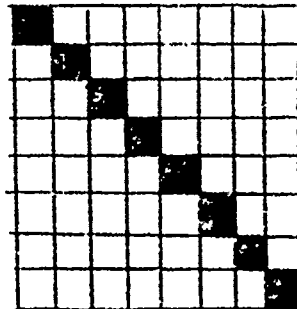
$$\delta([M,N],g) = [T]$$

$$\delta([M],f) = [T]$$

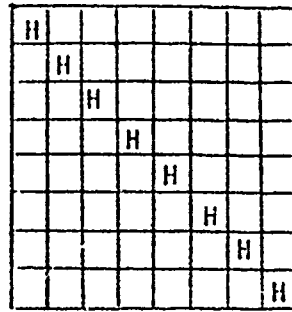
$$\delta([T],h) = [\phi]$$

The algorithm of highway recognition by the syntax-controlled method was implemented and the experiments were performed by the deterministic finite-state automata M_1' and M_2' . An illustrative example is given here to describe the operations of this algorithm.

Example: An image window from a LANDSAT image (visible bands) is processed by the transformation process and the transformed window is shown as:



The window pattern is encoded as the string dddd. The finite-state automaton, M_2' , works as follows: the input string is analyzed from the starting symbol [S]. The transition rule $\delta([S],d) = [B,D]$ is chosen by scanning the present states and inputs of all the transition rules of M_2' . Then a searching of the state [B,D] is performed on the present state of the transition rules of M_2' . The transition rule $\delta([B,D],d) = [K,J]$ is located and this rule leads to the next state [K,J]. Thereafter, the rule $\delta([K,J],d) = [T]$ is applied to obtain the next state [T] and the partial part (ddd) of the string (dddd) is recognized. The rule $\delta([T],d) = [\phi]$ is successfully found in the set of transition rules of M_2' . Thus, the string dddd is accepted by this automaton. The window pattern is recognized. The pattern is a highway which lies from northwest to southeast. The recognized pattern is:



The experiments of highway recognition by the automaton M_1' and M_2' were conducted on the same set of images. The experimental results from the automaton, M_2' , are more accurate than those of M_1' but the CPU time is slightly longer than that of the automaton, M_1' . Since the grammar inferred by $k=2$ describes patterns more precisely than the grammar inferred by $k=1$, the slight difference in CPU time between these two automata, M_1' and M_2' is acceptable. For example, the CPU time for automaton, M_1' on highway recognition of a 96 x 96 LANDSAT image is 20 seconds on the IBM 360/67 computer. The automaton, M_2' , analyzing the same image takes 21 seconds on the same computer. Thus, the grammar, $k=2$, is suggested as the better highway grammar. The grammars of the cases $k=3$ and $k=4$ are not suggested, because of the increase in computer processing time to infer these grammars and the increase in CPU time for the image recognition by the automata, M_3' and M_4' .

3.2.3. Experimental Results on Highway and River Recognition from LANDSAT Images

The computer experiments on highways and rivers were conducted in different areas. The highway recognition by the syntax-controlled method on the image (Figure 3.3(a)) is given in Figure 3.12. The area of Figure 3.12 is the same as that marked on Figure 3.3(d). The computer result shows that highways 1-65 and 1-465 were successfully recognized. The

HIGHWAY RECOGNITION RESULT

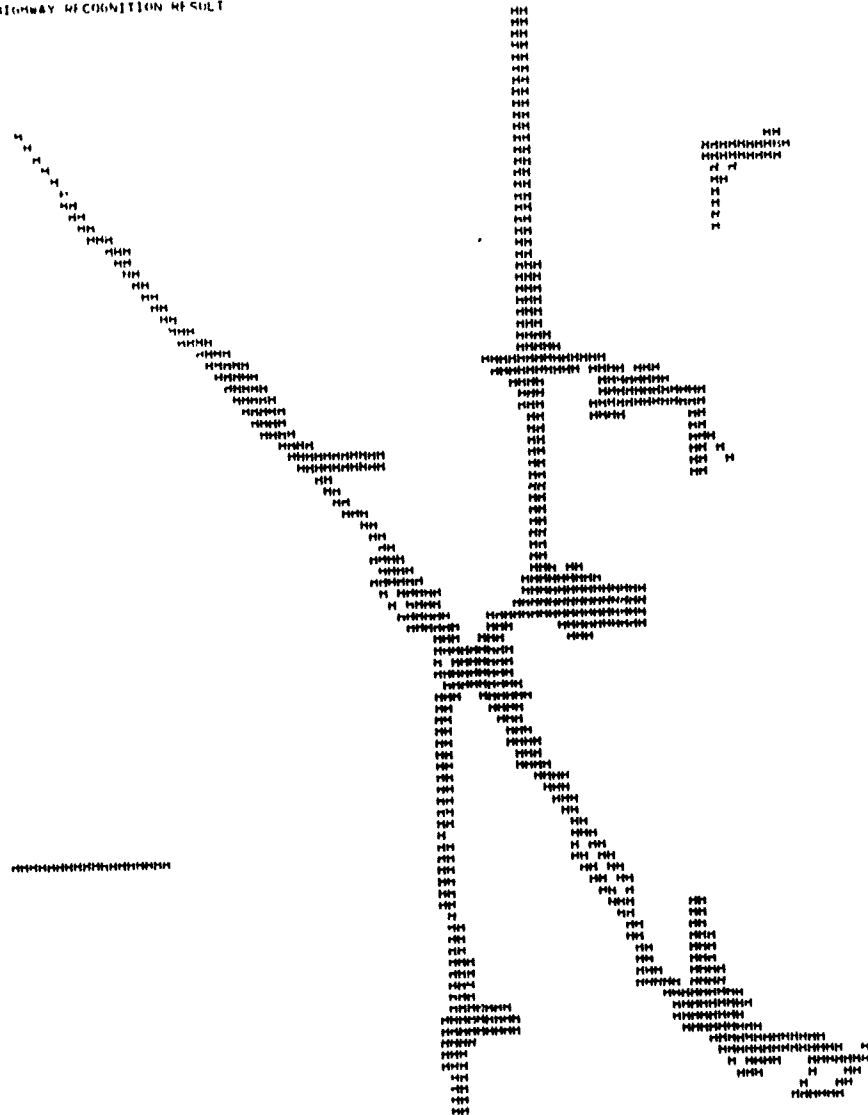


Figure 3.12. Highway recognition result by the syntax-controlled method on Figure 3.3(a).

RIVER RECOGNITION RESULT



Figure 3.13. River recognition result by syntax-controlled method on Figure 3.3(a).

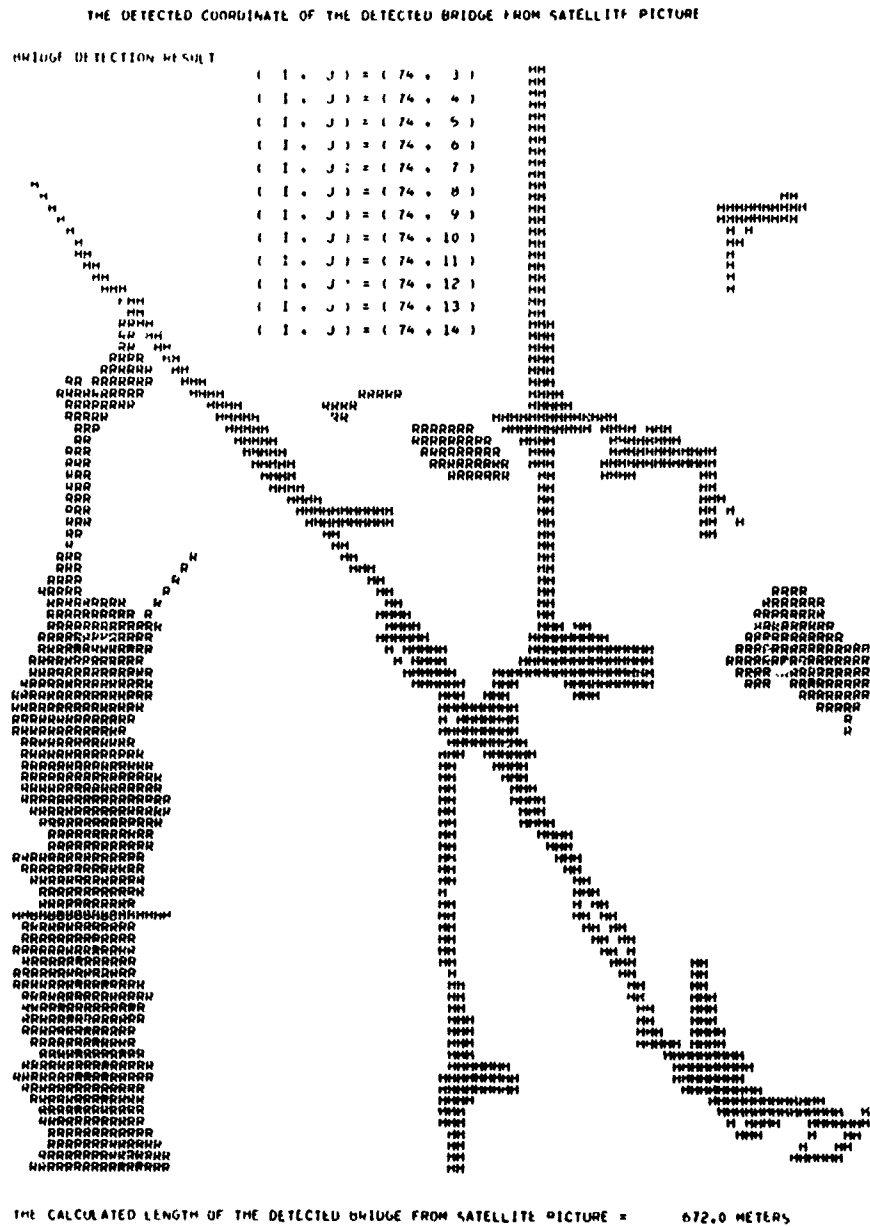


Figure 3.14. Bridge recognition result by syntax-controlled method on Figure 3.3(a).

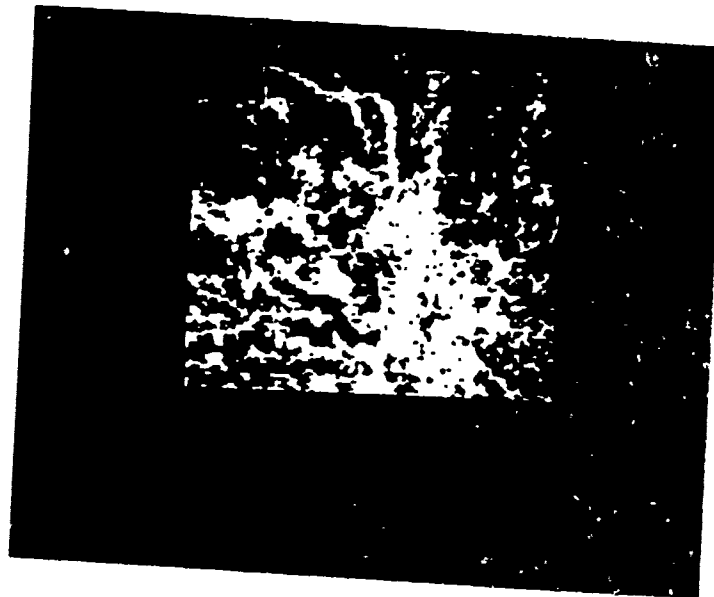


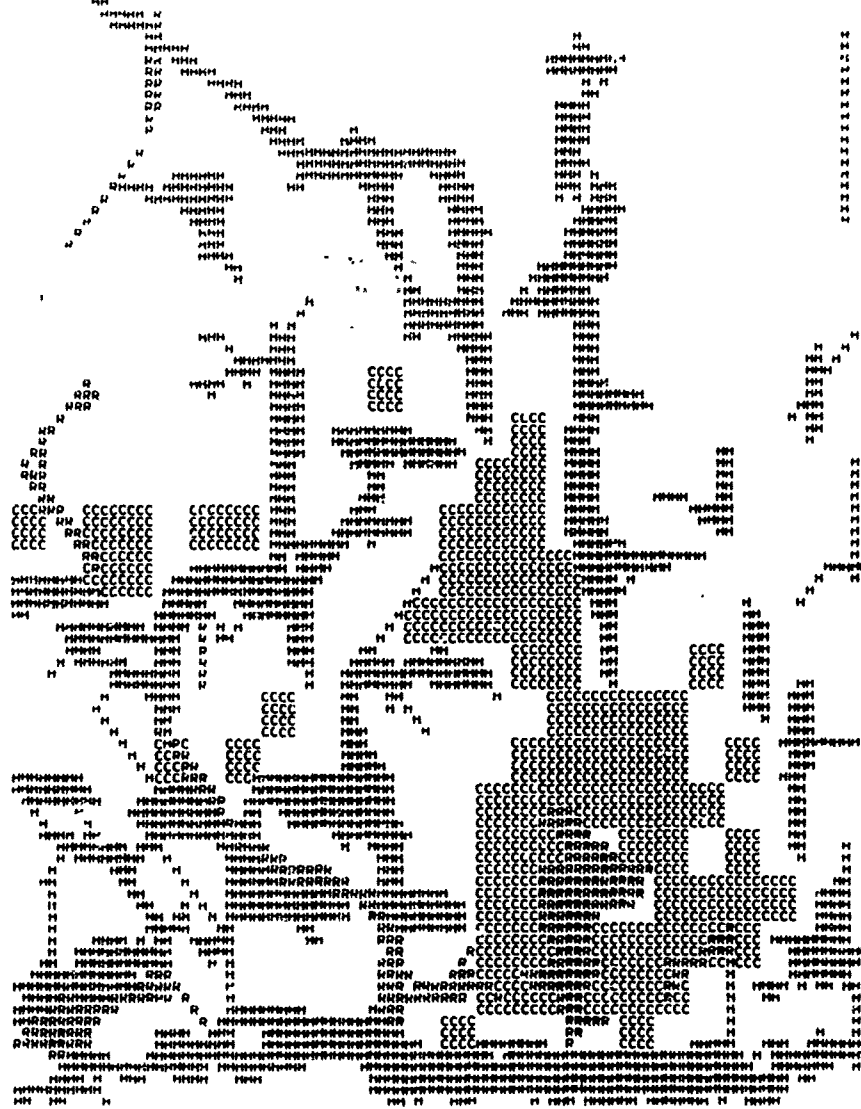
Figure 3.15(a). Satellite image of downtown Indianapolis, Indiana.

COMMERCIAL/INDUSTRIAL AREA RECOGNITION RESULT



Figure 3.15(b). Commercial/industrial area recognition by syntax-controlled method on Figure 3.15(a).

LAND USE CLASSIFICATION RESULT



THE AREA OF COMMERCIAL/INDUSTRIAL AREA = 13.10 SQUARE KILOMETERS
 THE CENTER OF THE COMMERCIAL/INDUSTRIAL AREA = (61.55)

Figure 3.15(c). Urban development information extraction result by syntax-controlled method on Figure 3.15(a).

INDIANAPOLIS Downtown Area

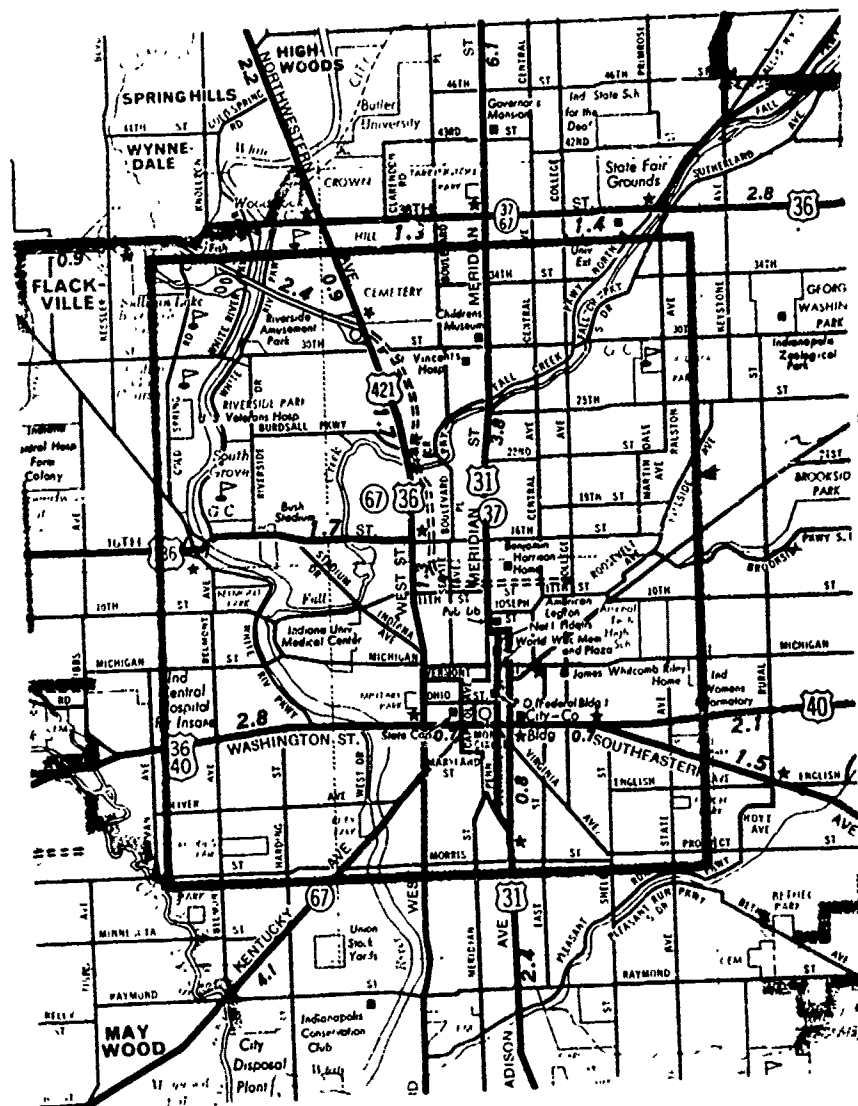


Fig. 3.15(a)

Figure 3.15(d). Indianapolis downtown map (marked area corresponds to Figure 3.15(a)).

Image is 96 x 96 in size. The CPU time for highway recognition by the syntax-directed method on the same image took 27 seconds compared with the 21 seconds here. The river recognition was performed on the same image shown in Figure 3.3(a). The results of river recognition by this syntax-controlled method is shown in Figure 3.13. The CPU processing time for this method is 12 seconds by the automaton M_2' ($k=2$ case) on the 96 x 96 image compared with 20 seconds for the syntax-directed method.

3.2.4. Experimental Results on Bridge and Commercial/Industrial Area Recognition from LANDSAT Images

Figure 3.14 shows the bridge recognition result by the syntax-controlled method. The recognizer is automaton M_2' ($k=2$ case). The length is automatically calculated as 672 meters which is the same result as that obtained by the syntax-directed method. The CPU time for bridge recognition by the syntax-controlled method is 39 seconds, whereas the syntax-directed method requires a longer CPU time, 42 seconds. The commercial/industrial area recognition was performed by the syntax-controlled method on an image of the downtown area of Indianapolis, Indiana, Figure 3.15(a). The result is shown in Figure 3.15(b). The size and center of the downtown area were automatically calculated from the 96 x 96 image frame. The computer processing time of the syntax-controlled method for this experiment was 43 seconds. The downtown map of Indianapolis is provided in Figure 3.15(d). The urban development information extraction is shown in Figure 3.15(c).

3.3. COMPARISON OF THE SYNTAX-DIRECTED AND SYNTAX-CONTROLLED METHODS

As has been stated earlier, the syntax-directed and syntax-controlled methods were implemented and experiments on highway, river, bridge, and

commercial/industrial area recognition were conducted on the IBM 360/67 computer in the Laboratory of Applications for Remote Sensing (LARS). The LOGICAL programming technique was used in both methods. Comparative studies were also carried out without LOGICAL programming. The one using logical programming saved 30% of the CPU time used by the other. Concerning computer memory space, there is another advantage to LOGICAL programming in that every pixel of the transformed image takes only one byte for storage. (Usually each pixel takes 4 bytes for storage of integer). Therefore, it was found in this experiment that the use of LOGICAL programming saved approximately 75% of the memory space required by the program requiring four bytes per pixel.

As stated above the CPU time for the syntax-directed method for highway recognition from a 96 x 96 image takes approximately 27 seconds. The CPU time for the syntax-directed method for river, bridge, and commercial/industrial area recognition from 96 x 96 images takes 20, 42, and 46 seconds, respectively. The syntax-controlled method was implemented by the same programming techniques as those used for the syntax-directed method. The CPU time for this second method for highway, river, bridge, and commercial/industrial area recognition from the 96 x 96 image were 20, 12, 38, and 41 seconds, respectively (for an automaton corresponding to a grammar by the $k=1$ inference procedure). For an automaton corresponding to the grammar by the $k=2$ inference procedure, the CPU time for highway, river, bridge, and commercial/industrial area recognition of the same images were 21, 12, 39 and 43 seconds, respectively. The comparative performances of the syntax-directed and syntax-controlled methods are listed in Table 3.1. It can be seen that the syntax-controlled method processes the same image for all the tasks faster than the syntax-

Table 3.1. CPU time performance comparison of syntax-directed and syntax-controlled methods.

Method \ Task	Highway	River	Bridge	Commercial/ Industrial
Syntax-Directed	27 sec.	20 sec.	42 sec.	46 sec.
Syntax-Controlled k=1	20 sec.	12 sec.	38 sec.	41 sec.
Syntax-Controlled k=2	21 sec.	12 sec.	39 sec.	43 sec.

directed method. As for the accuracy, the syntax directed and syntax-controlled methods have the same level of high accuracy in the experiments.

In general, the syntax-directed methods, commonly known as the template-matching method, has the advantage of faster software development time because the implementation is less complicated than for the syntax-controlled method. However, the disadvantage of the syntax-directed method is that the recognition depends on a set of limited template patterns. Once the pattern does not match correctly with one of the templates, then the pattern is rejected. The only way to increase the recognition capability is by adding more templates in the set of template windows. These added templates increase the need for computer storage and the CPU time for each matching operation. The syntax-controlled method has the advantage of fast computer processing time for program execution once the more complicated programming is completed which means a saving of CPU time in processing the image every time compared with that of the syntax-directed method. Another advantage of the syntax-controlled method of finite-state string grammar is that the grammatical inference can be fully computer automated. Thus, the grammar is more realistic and precise in describing the patterns than is the grammar for syntax-directed method. The CPU time for the grammatical inference of a highway grammar is 9 seconds for the case of $k=1$ and 15 seconds for the case of $k=2$ in the k -tail inference method of the syntax-controlled method, as compared with the grammatical inference procedure for the syntax-directed method in which, because the procedure is an interactive process, the CPU time is about 60 seconds.

In conclusion, the syntax-controlled method for highway, river, bridge, and commercial/industrial area recognition from LANDSAT images is an effective technique for image recognition. The results from such applications can contribute to urban development planning, and to military reconnaissance.

CHAPTER 4

IMAGE SEGMENTATION USING A TREE GRAMMAR APPROACH

The syntactic method for image segmentation was developed by a tree grammar approach. The reasons that a tree grammar approach is proposed for image segmentation are; first, tree languages are very descriptive and second, the tree grammar analysis offers a natural high-dimensional generalization of strings. Since the boundary patterns of an image are usually two dimensional and tree grammar is more convenient for describing high-dimensional objects than a string grammar, the tree grammar is used to describe the boundaries of the homogeneous segments of an image. In addition, the high recognition capability of tree automaton, corresponding to the tree grammar, and the hierarchical nature of scenes make the tree grammar approach very attractive for image segmentation.

A syntactic approach to image segmentation was investigated which involves two levels of processing. The first level, referred to as pre-processing and primitive extraction, consists of two steps (1) texture region primitive extraction, and (2) boundary primitive extraction. The second level, which is the syntactic analysis, requires tree grammar inference to describe the boundaries of homogeneous regions. A block diagram of the system for image segmentation is shown in Figure 4.1. The process of tree grammar analysis utilizes the corresponding tree automaton from the inferred tree grammar to process the primitive extracted image. Then this image is segmented.

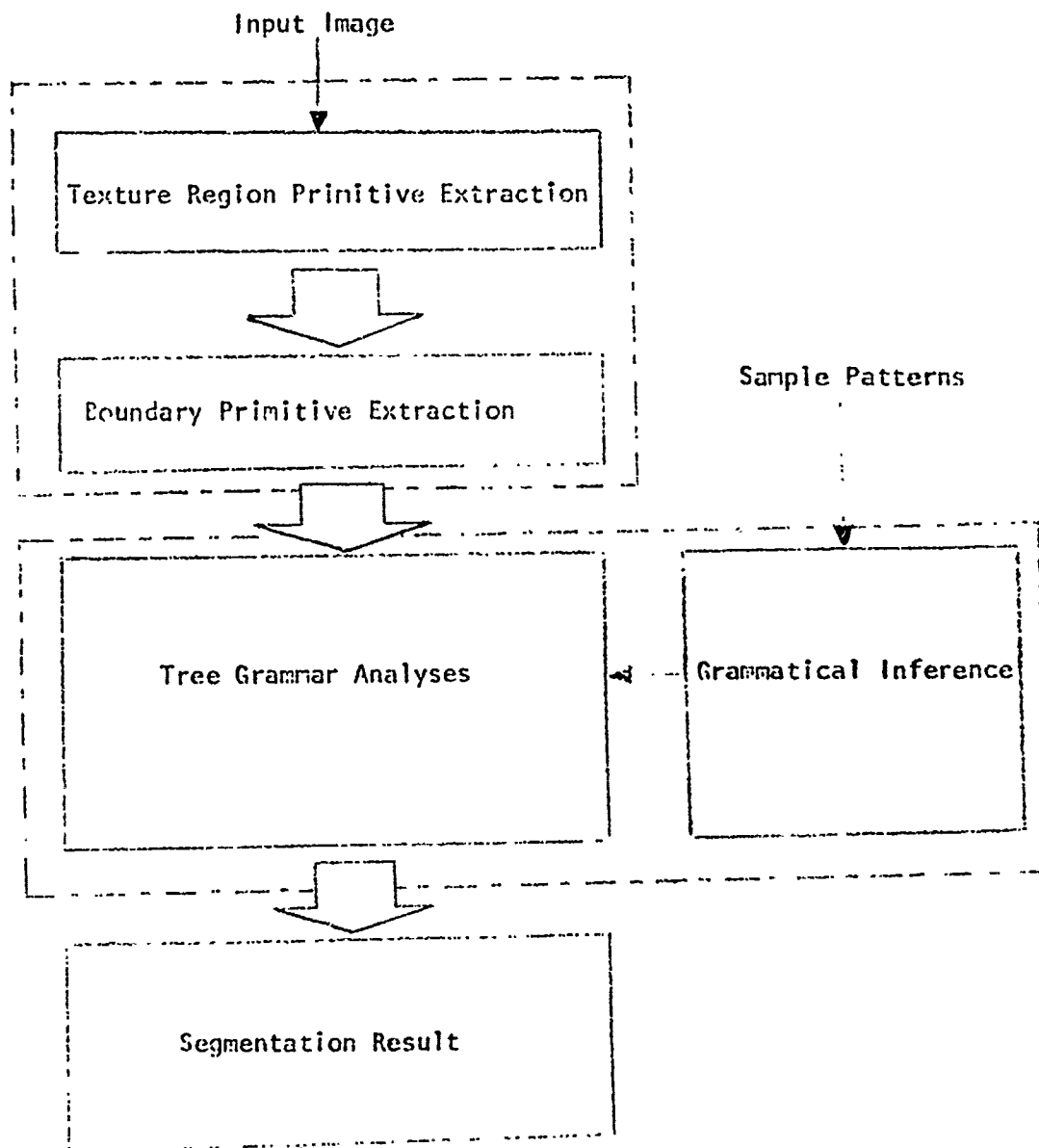


Figure 4.1. Block diagram of the system for image segmentation.

4.1 SYNTACTIC IMAGE SEGMENTATION ALGORITHM (SISA)

Several approaches to the use of texture information in image analysis have recently been developed [72,73]. However, these techniques have generally been applied to terrain classification following segmentation and not to the segmentation problem itself. In our approach to image segmentation, the texture information and boundary structures are analyzed to obtain the image segmentation result. The input to the system is the digitized image. The texture region primitive extraction performs the texture analysis and extracts the texture region primitive. Then, the boundary primitive extraction extracts the boundary primitives from the result of the texture region primitive extraction. The result from the boundary primitive extraction is then modeled by the tree grammar which is inferred by the tree grammar's grammatical inference procedure. Each part of the image segmentation system is discussed in detail below.

4.1.1 Inference of Tree Grammars

The concept of tree grammar analysis is herein described by defining and studying tree grammars and tree automata [2].

Definition 1:

A pattern grammar G is a four tuple $G = (V_H, V_T, P, S)$ where

V_H is a set of non-terminals or subpatterns.

V_T is a set of terminals or pattern primitives.

$S \in V_H$ is the start symbol, and

P is a set of syntax rules or productions in the form of $\alpha \rightarrow \beta$.

$\alpha \in (V_H \cup V_T)^* V_H (V_H \cup V_T)^*$, $\beta \in (V_H \cup V_T)^*$

V^* is the set of all possible sequences of symbols in V including the empty sequence λ .

By applying productions in P successively, starting from S , the strings or sentences of terminals generated by the production in P are the language generated by the grammar G , $L(G)$.

Definition 2:

Let N^+ be the set of strictly positive integers.

Let U be the universal tree domain (the free semigroup with identity element "0" generated by N^+ and a binary operation " \cdot ") [2,95].

Definition 3:

A ranked alphabet is a pair $\langle \Sigma, r \rangle$ where Σ is a finite set of symbols and $r: \Sigma \rightarrow N = N^+ \cup \{0\}$. For $a \in \Sigma$, $r(a)$ is called the rank of a .

Definition 4:

A tree over Σ (i.e., over $\langle \Sigma, r \rangle$) is a function $\alpha: D \rightarrow \Sigma$ such that D is a tree domain and $r[\alpha(a)] = \max\{i \mid a \cdot i \in D\}$. The domain of a tree, α , is denoted by $D(\alpha)$. Let T_Σ be the set of all trees over Σ .

Definition 5:

Let α be a tree and a be a member of $D(\alpha)$, $\alpha|_a$, a subtree of α at a is defined as $\alpha|_a = \{(b, x) \mid (a \cdot b, x) \in \alpha\}$.

Definition 6:

A regular tree grammar G_t over $\langle V_T, r \rangle$ is a grammar $G_t = (V, r', P, S)$ satisfying the following conditions:

- (i) $\langle V, r' \rangle$ is a finite ranked alphabet such that $V_T \subseteq V$ and $r'|_{V_T} = r$, V_T and $V - V_T = V_N$ are the same as in Definition 1.
- (ii) P is a finite set of productions of the form $\phi \rightarrow \psi$ where $\phi, \psi \in T_V$ (T_V is a set of trees over $\langle V, r' \rangle$).
- (iii) S is a finite subset of T_V .

Definition 7

A tree grammar $G_t = (V, r, P, S)$ is expansive if and only if each production in P is of the form

$$X_0 \rightarrow \begin{array}{c} x \\ \swarrow \quad \searrow \\ X_1 \quad \dots \quad X_r \end{array}$$

where $x \in V_T$ and $X_1, X_2, \dots, X_r \in V_N (= V - V_T)$ are non-terminal symbols.

Theorem 1: For each regular tree grammar, G_t , one can effectively construct an equivalent expansive grammar G_t' , that is, $L(G_t') = L(G_t)$ [2].

Definition 8

A tree automaton over Σ is a $(k+2)$ -tuple

$$M_t = (Q, f_1, \dots, f_k, F)$$

where (i) Q is a finite set of states; (ii) for each $i, 1 \leq i \leq k$, f_i is a relation on $Q^{r(\sigma_i)} \times Q$, $\sigma_i \in \Sigma$, that is $f_i : Q^{r(\sigma_i)} \rightarrow Q$; and (iii) $F \subseteq Q$ is a set of final states.

Definition 9

The response relation P of a tree automaton M_t is defined as (i) if $\sigma \in \Sigma_0$, $p(\sigma) = X$ (X belongs to final state set F of the tree automaton) if and only if $f_\sigma = X$, that is, $p(\sigma) = f_\sigma$; (ii) if $\sigma \in \Sigma_n$, $n \geq 0$, $p(\sigma, X_0, \dots, X_{n-1}) = X$ (X_0, \dots, X_{n-1} are non-terminals) if and only if there exists X_0', \dots, X_{n-1}' such that $f_\sigma(X_0', \dots, X_{n-1}') = X$.

Definition 10

$T(M_t) = \{\alpha \in T_\Sigma \mid \text{there exists } X \in F \text{ such that } p(\alpha) = X\}$ is called the set of trees accepted by M_t .

Theorem 2: For every regular tree grammar, G_t , one can effectively construct a tree automaton M_t , such that $T(M_t) = L(G_t)$ [2].

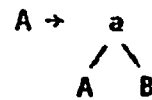
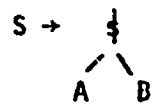
The construction procedure is summarized as follows:

- (i) Obtain an expansive tree grammar $G_t' = (V', r, P', S)$ for the given regular tree grammar $G_t = (V, r, P, S)$ over alphabet V_T .

- (ii) The equivalent (nondeterministic) tree automaton is

$M_t = (Q, f_1, \dots, f_k, F)$ where $Q = \{X_0, X_1, \dots, X_n\} \supset F$
 $f_x(X_1, \dots, X_n) = X_0$ if $X_0 \rightarrow x X_1, \dots, X_n$ is in P' , $x \in V_T$,
 $X_1, \dots, X_n \in Q$, and $f_x(X_x) = X_0$ if $X_0 \rightarrow x$ is in P' . $x \in V_T$,
 $X_1, \dots, X_n \in Q$, F is the set of final state. $F = \{X_x | x \in V_T \text{ and } f_x(X_x) = X_0\}$.

An illustrative example is provided here to show the procedure of constructing the tree automaton. The tree grammar is $G_t = (V, r, P, S)$, where $V = \{S, \frac{1}{2}, a, b, A, B\}$, $\frac{1}{2}$ is the starting node or root of the tree, $V_T = \{a, b, \frac{1}{2}\}$, $r(a) = \{2, 0\}$, $r(b) = \{0\}$, $r(\frac{1}{2}) = \{2\}$, and P :



$A \rightarrow a$

$B \rightarrow b$


The procedure of construction for tree automaton is as follows:

- (i) Obtain an expansive tree grammar $G_t' = (V', r, P', S)$.


Since G_t is an expansive tree grammar, thus $G_t' = G_t = (V, r, P, S)$ for this example grammar.

- (ii) The tree automaton is $M = (Q, f_{\frac{1}{2}}, f_a, f_b, F)$. The Q , $f_{\frac{1}{2}}$, f_a , f_b , and F are constructed as follows. Since the grammar

rule is $S \rightarrow \xi$, the relation is obtained as $f_\xi(q_A, q_B) = q_S$.



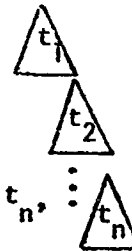
The relation $f_a(q_A, q_B) = q_A$ is obtained from grammar rule $A \rightarrow a$. The relation $f_a(q_a) = q_A$ is obtained from grammar



rule $A \rightarrow a$, and $f_b(q_b) = q_B$ is obtained from grammar rule $B \rightarrow b$. The tree automaton is thus constructed as $M_t = \{q_S, q_A, q_B, q_a, q_b\}$, the relation f 's are f_ξ, f_a, f_b , and the final state set $F = \{q_a, q_b\}$.

In order to model a language more realistically or to describe a class of patterns more precisely, it is expected that the grammar used can be directly inferred from a set of sample sentences or a set of sample patterns. This subject of "learning" a grammar based on a set of sample sentences is called grammatical inference. A tree grammar inference procedure is briefly reviewed [20] below:

- (i) Represent each sample tree α_i as

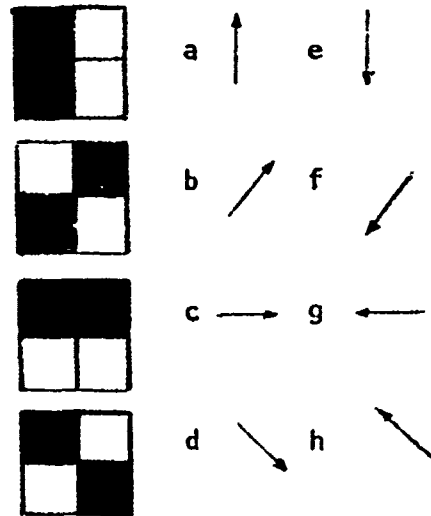


where any of the subtrees t_1, t_2, \dots, t_n consists of repetitive substructures.

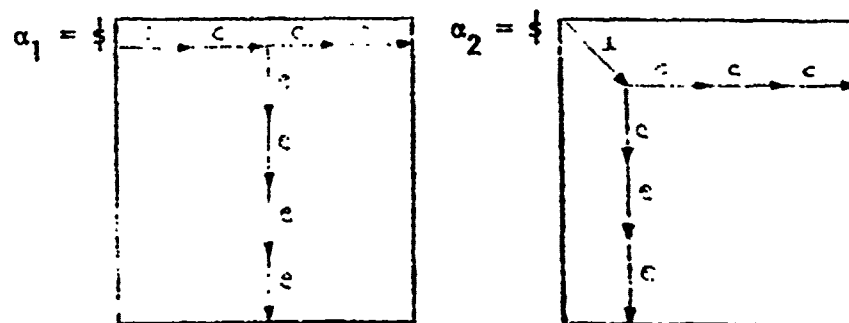
- (ii) Starting from the root, determine subtrees with depth one for each sample tree α_i , excluding the subtrees having repetitive substructures.
- (iii) Attach non-terminal symbols to nodes and construct an expansive tree grammar G_i for α_i .
- (iv) The inferred tree grammar for the complete sample set will then be

$$G_t = \bigcup_i G_i$$

The purpose of the inferred tree grammar is to describe the boundaries of the image segments. The primitives for those patterns are:

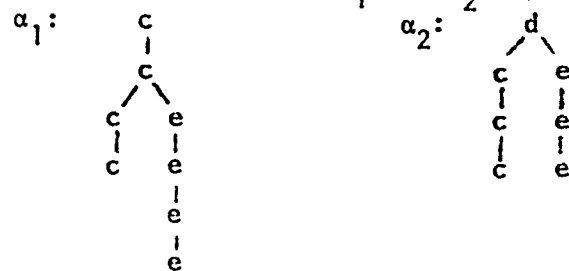


and the window size is chosen as an 8x8 array of pixels. The positive samples are those patterns starting from a primitive followed by, at most, three branches. The negative samples are those patterns in which there is no boundary line or those having singular primitives or pixels. The sample patterns are listed in Appendix A. Applying the tree grammatical inference procedure [20], a set of tree grammars is inferred to describe the boundary structures. An illustrative example is given here to show the grammatical inference procedure of a tree grammar. For example, the sample patterns are given as follows.

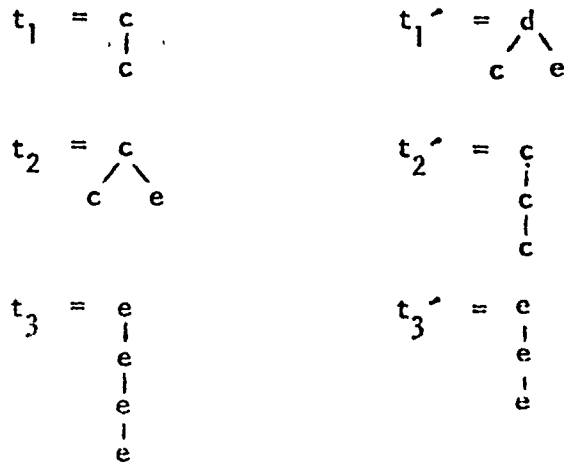


The grammatical inference procedure works as follows: (The flow chart of the tree grammar inference procedure is provided in Appendix G).

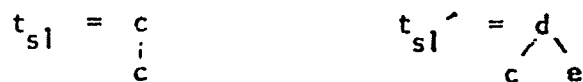
Step 1. Represent sample trees α_1 and α_2 as,



The subtrees of α_1 and α_2 are t_1 , t_2 , and t_3 for α_1 and t_1' , t_2' , and t_3' for α_2 .



Step 2. Determine subtrees with depth one.



$$t_{s2} = \begin{array}{c} c \\ / \quad \backslash \\ c \quad e \end{array}$$

$$t_{s2}' = \begin{array}{c} c \\ | \\ c \end{array}$$

$$t_{s3} = \begin{array}{c} e \\ | \\ e \end{array}$$

$$t_{s3}' = \begin{array}{c} e \\ | \\ e \end{array}$$

Step 3. Attach nonterminal symbols to nodes and construct an expansive tree grammar G_1 for α_1 and G_2 for α_2 . ($\frac{1}{2}$ starting symbol)

$$G_1: \quad S \rightarrow \frac{1}{2} \\ \quad \quad | \\ \quad \quad A_1$$

$$G_2: \quad S \rightarrow \frac{1}{2} \\ \quad \quad | \\ \quad \quad A_2$$

$$A_1 \rightarrow c \\ \quad \quad | \\ \quad \quad A_1$$

$$A_2 \rightarrow d \\ \quad \quad / \quad \backslash \\ \quad \quad A_1 \quad A_2$$

$$A_1 \rightarrow c \\ \quad \quad / \quad \backslash \\ \quad \quad A_1 \quad A_2$$

$$A_2 \rightarrow e \\ \quad \quad | \\ \quad \quad A_2$$

$$A_1 \rightarrow c$$

$$A_2 \rightarrow e$$

Step 4. The inferred grammar is G_t which is the union set of the grammar set G_1 and G_2 of Step 3.

The tree grammar is obtained from the grammatical inference procedure. The tree grammar G_t is as follows.

$$G_t = (V, r, P, S)$$

where $V = \{S, \$, A_1, A_2, A_3, A_4, A_5, A_6, A_7, a, b, c, d, e, f, g, h\}$

$$r(a) = \{1, 0\} \quad r(b) = \{1, 0\} \quad r(c) = \{2, 1, 0\} \quad r(d) = \{3, 2, 1, 0\}$$

$$r(e) = \{3, 2, 1, 0\} \quad r(f) = \{2, 1, 0\} \quad r(g) = \{2, 1, 0\} \quad r(h) = \{1, 0\}$$

$$r(\$) = \{2, 1\}$$

$$V_T = \{\uparrow a, \times b, \rightarrow c, \times d, \uparrow e, \times f, \leftarrow g, \times h, \$\}$$

and grammar rules P:

$$S \rightarrow \begin{array}{c} \$ \\ \swarrow \quad \searrow \\ A_1 \quad A_2 \end{array} \quad S \rightarrow \begin{array}{c} \$ \\ | \\ A_1 \end{array} \quad S \rightarrow \begin{array}{c} \$ \\ | \\ A_6 \end{array} \quad S \rightarrow \begin{array}{c} \$ \\ | \\ A_2 \end{array} \quad S \rightarrow \begin{array}{c} \$ \\ | \\ A_7 \end{array}$$

$$A_1 \rightarrow \begin{array}{c} e \\ \swarrow \quad \searrow \\ A_1 \quad A_2 \end{array}$$

$$A_2 \rightarrow \begin{array}{c} c \\ \swarrow \quad \searrow \\ A_1 \quad A_2 \end{array}$$

$$A_1 \rightarrow \begin{array}{c} c \\ \swarrow \quad \searrow \\ A_1 \quad A_2 \end{array}$$

$$A_2 \rightarrow \begin{array}{c} d \\ \swarrow \quad \searrow \\ A_1 \quad A_2 \end{array}$$

$$A_1 \rightarrow \begin{array}{c} f \\ \swarrow \quad \searrow \\ A_1 \quad A_2 \end{array}$$

$$A_2 \rightarrow \begin{array}{c} g \\ \swarrow \quad \searrow \\ A_1 \quad A_2 \end{array}$$

$$A_1 \rightarrow \begin{array}{c} d \\ \swarrow \quad | \quad \searrow \\ A_3 \quad A_4 \quad A_5 \end{array}$$

$$A_2 \rightarrow \begin{array}{c} f \\ \swarrow \quad \searrow \\ A_1 \quad A_2 \end{array}$$

$$A_1 \rightarrow \begin{array}{c} e \\ \swarrow \quad | \quad \searrow \\ A_3 \quad A_4 \quad A_5 \end{array}$$

$$A_3 \rightarrow \begin{array}{c} b \\ | \\ A_3 \end{array}$$

$$A_3 \rightarrow \begin{array}{c} d \\ | \\ A_3 \end{array}$$

$$A_3 \rightarrow \begin{array}{c} e \\ | \\ A_3 \end{array}$$

$$A_3 \rightarrow \begin{array}{c} f \\ | \\ A_3 \end{array}$$

$$A_3 \rightarrow \begin{array}{c} g \\ | \\ A_3 \end{array}$$

$$A_3 \rightarrow \begin{array}{c} c \\ | \\ A_3 \end{array}$$

$$A_3 \rightarrow \begin{array}{c} h \\ | \\ A_3 \end{array}$$

$$A_4 \rightarrow \begin{array}{c} b \\ | \\ A_4 \end{array}$$

$$A_4 \rightarrow \begin{array}{c} d \\ | \\ A_4 \end{array}$$

$$A_4 \rightarrow \begin{array}{c} e \\ | \\ A_4 \end{array}$$

$$A_4 \rightarrow \begin{array}{c} a \\ | \\ A_4 \end{array}$$

$$A_4 \rightarrow \begin{array}{c} f \\ | \\ A_4 \end{array}$$

$$A_4 \rightarrow \begin{array}{c} g \\ | \\ A_4 \end{array}$$

$$A_4 \rightarrow \begin{array}{c} c \\ | \\ A_4 \end{array}$$

$$A_4 \rightarrow \begin{array}{c} h \\ | \\ A_4 \end{array}$$

$$A_5 \rightarrow \begin{array}{c} b \\ | \\ A_5 \end{array}$$

$$A_5 \rightarrow \begin{array}{c} d \\ | \\ A_5 \end{array}$$

$$A_5 \rightarrow \begin{array}{c} e \\ | \\ A_5 \end{array}$$

$$A_1 \rightarrow \begin{array}{c} e \\ | \\ A_4 \end{array}$$

$$A_5 \rightarrow \begin{array}{c} f \\ | \\ A_5 \end{array}$$

$$A_5 \rightarrow \begin{array}{c} g \\ | \\ A_5 \end{array}$$

$$A_5 \rightarrow \begin{array}{c} c \\ | \\ A_5 \end{array}$$

$$A_5 \rightarrow \begin{array}{c} h \\ | \\ A_5 \end{array}$$

$$A_6 \rightarrow \begin{array}{c} e \\ | \\ A_6 \end{array}$$

$$A_6 \rightarrow \begin{array}{c} e \\ | \\ A_1 \end{array}$$

$$A_7 \rightarrow \begin{array}{c} c \\ | \\ A_7 \end{array}$$

$$A_7 \rightarrow \begin{array}{c} c \\ | \\ A_1 \end{array}$$

$$A_1 \rightarrow \begin{array}{c} e \\ | \\ A_1 \end{array}$$

$$A_1 \rightarrow \begin{array}{c} c \\ | \\ A_1 \end{array}$$

$$A_1 \rightarrow \begin{array}{c} f \\ | \\ A_1 \end{array}$$

$$A_2 \rightarrow \begin{array}{c} b \\ | \\ A_2 \end{array}$$

$$A_2 \rightarrow \begin{array}{c} c \\ | \\ A_2 \end{array}$$

$$A_2 \rightarrow \begin{array}{c} d \\ | \\ A_2 \end{array}$$

$$A_2 \rightarrow \begin{array}{c} g \\ | \\ A_2 \end{array}$$

$$A_2 \rightarrow \begin{array}{c} f \\ | \\ A_2 \end{array}$$

$$A_1 \rightarrow e$$

$$A_1 \rightarrow c$$

$$A_1 \rightarrow f$$

$$A_2 \rightarrow b$$

$$A_2 \rightarrow c$$

$$A_2 \rightarrow d$$

$$A_2 \rightarrow g$$

$$A_2 \rightarrow f$$

$$A_3 \rightarrow b$$

$$A_3 \rightarrow d$$

$$A_3 \rightarrow e$$

$$A_3 \rightarrow e$$

$$A_3 \rightarrow f$$

$$A_3 \rightarrow g$$

$$A_3 \rightarrow c$$

$$A_3 \rightarrow h$$

$$A_4 \rightarrow b$$

$$A_4 \rightarrow d$$

$$A_4 \rightarrow e$$

$$A_4 \rightarrow e$$

$$A_4 \rightarrow f$$

$$A_4 \rightarrow g$$

$$A_4 \rightarrow c$$

$$A_4 \rightarrow h$$

$$A_5 \rightarrow b$$

$$A_5 \rightarrow d$$

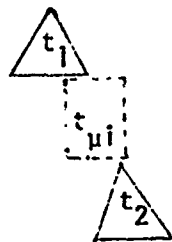
$$A_5 \rightarrow e$$

$$\begin{array}{llll}
 A_5 \rightarrow f & A_5 \rightarrow g & A_5 \rightarrow c & A_5 \rightarrow h \\
 A_6 \rightarrow e & A_7 \rightarrow c & A_k \rightarrow a &
 \end{array}$$

A transformational grammar is a set of grammatical rules for transforming a pattern from one form to another [2,21]. A line smoothing technique can be designed by a transformational grammar. Here we introduce the tree transformational grammar for the line smoothing technique. The concept of the syntactic line smoothing technique is as follows: Irregularities are usually caused by the digitizer, noisy patterns, and so forth. These are in forms such as the zig-zagging of the line patterns. The tree transformational grammar evaluates the contextual information of the patterns. If the context of the pattern satisfies the transformational grammar, that pattern is transformed into a smoother pattern. By this syntactic line smoothing technique, the zig-zagging of lines is smoothed. Actually, tree transformational grammar is a universal method for line smoothing of any pictorial data.

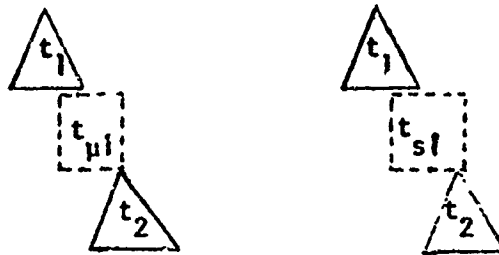
The process grammatical inference scheme for tree transformational grammar is as follows:

- (1) Represent each non-smooth pattern as



where t_1 and t_2 are the predecessor and successor subtrees of the $t_{\mu i}$ subtree.

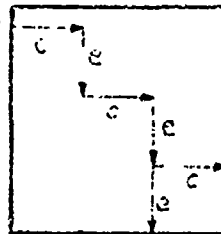
- (2) Based on the relationship of t_1 , t_2 , and $t_{\mu i}$, interchange $t_{\mu i}$ with a smoother pattern t_{si} , then obtain tree transformational grammar G_i for $t_{\mu i}$, as:



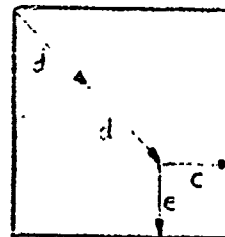
- (3) The inferred tree transformational grammar for the complete sample set will then be $G_t = \bigcup_i G_i$

For example:

non-smooth pattern (6x6) ‡

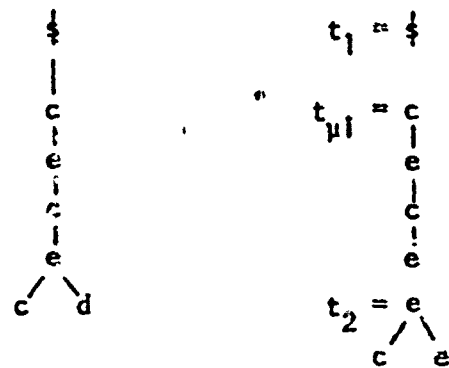


smoothed pattern (6x6) ‡

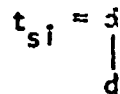


The grammatical inference scheme for tree transformational grammar works as follows.

Step 1. Represent non-smooth pattern as

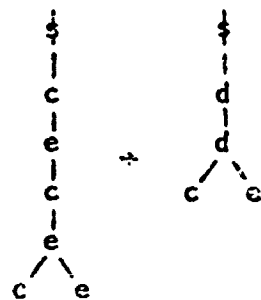


Step 2. The smoother pattern t_{si} of $t_{\mu i}$ is



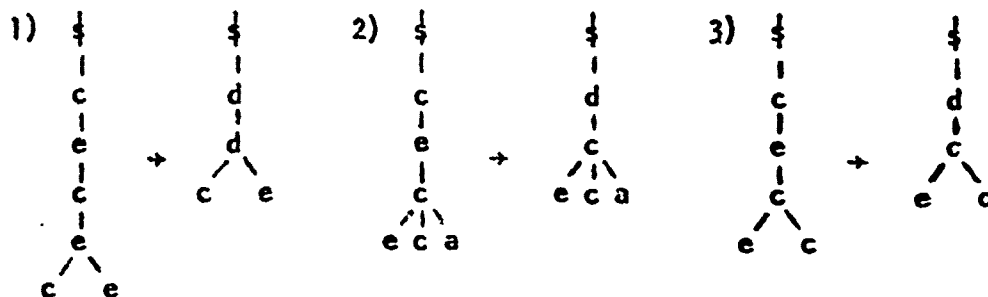
and the t_1 and t_2 of the smoother pattern are the same as for the non-smooth pattern.

Step 3. The inferred transformational grammar G_1 is

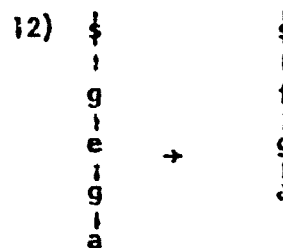
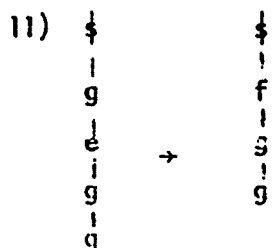
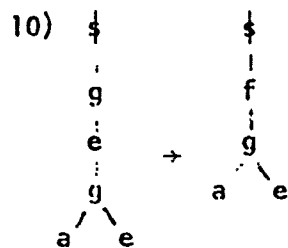
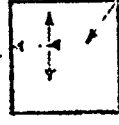
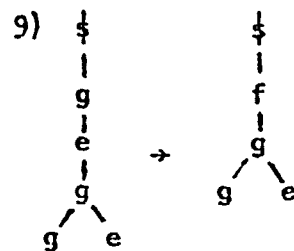
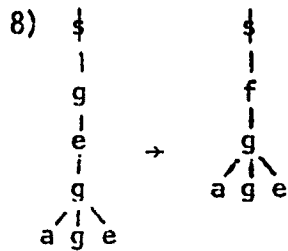
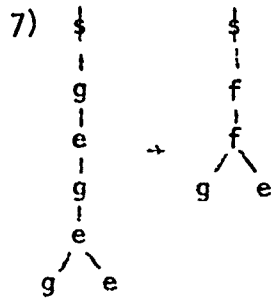
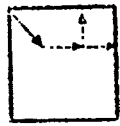
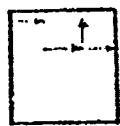
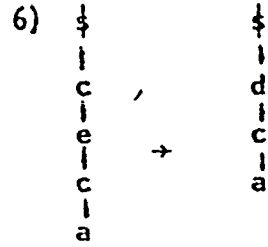
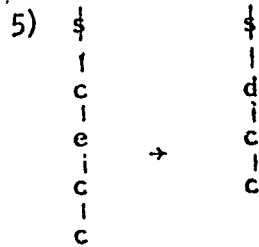
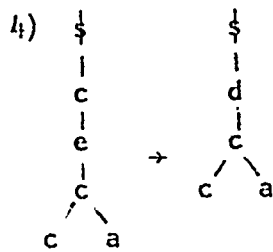
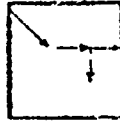
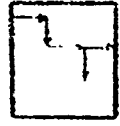
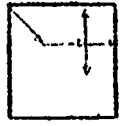
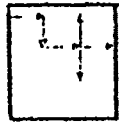
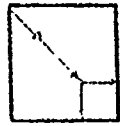
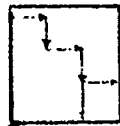


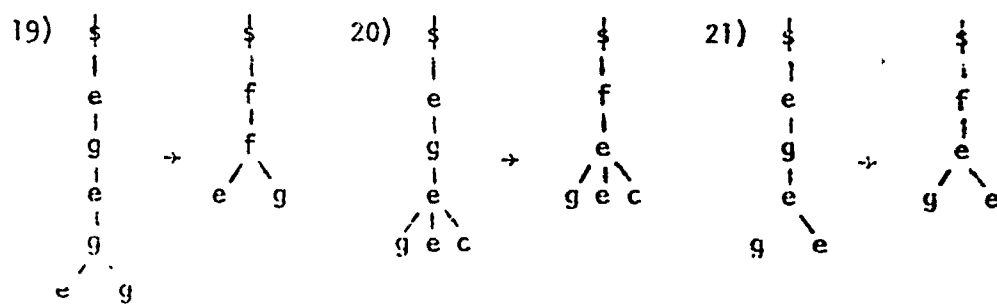
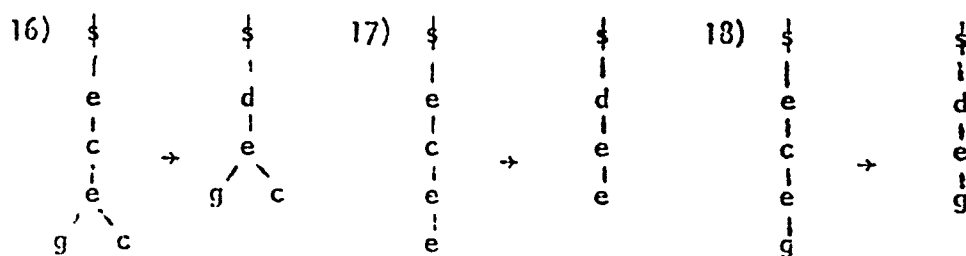
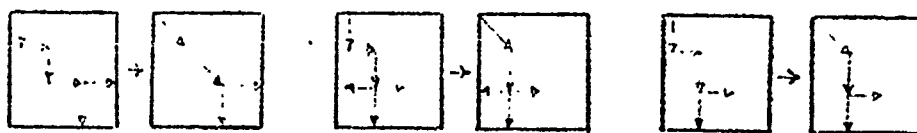
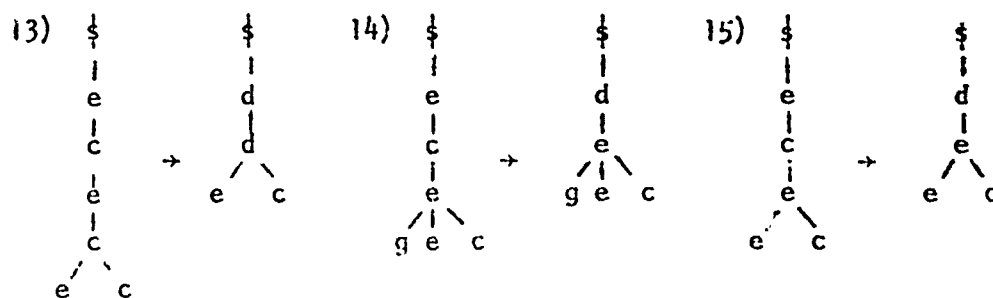
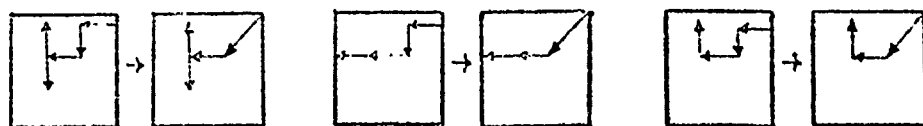
The tree transformational grammar G_t is the union of the grammar G_1 .

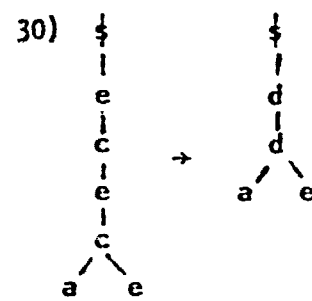
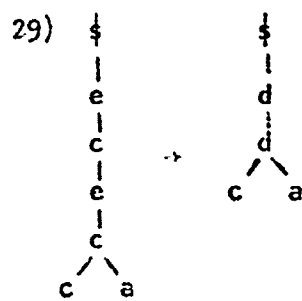
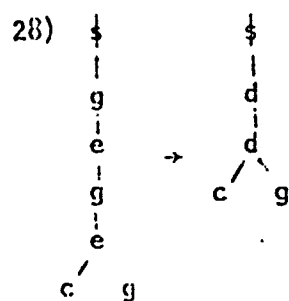
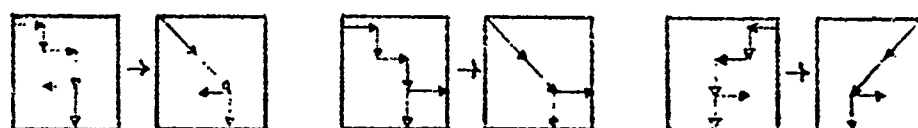
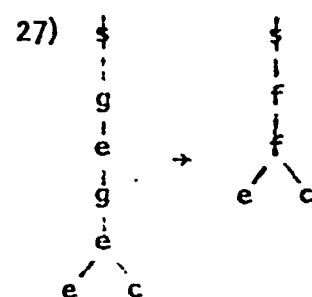
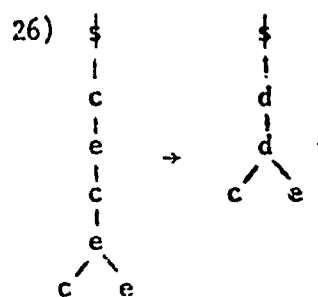
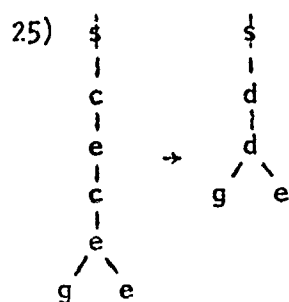
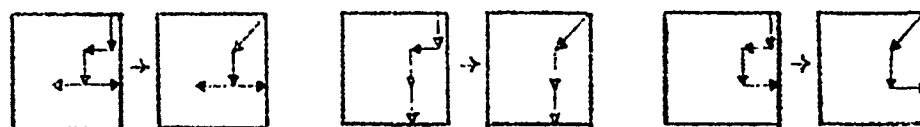
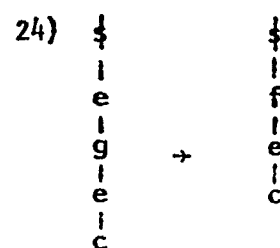
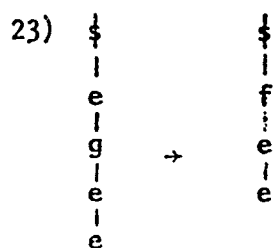
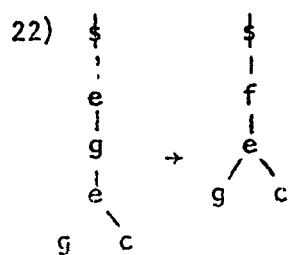
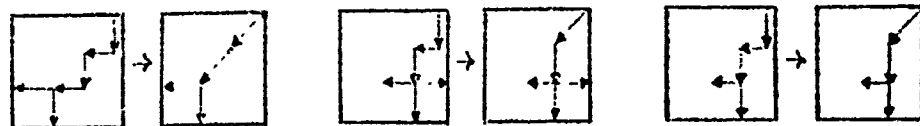
For the syntactic line smoothing technique in section 4.1.3, a tree transformational grammar is inferred to reduce irregularities and smooth the patterns. The grammar is as follows:

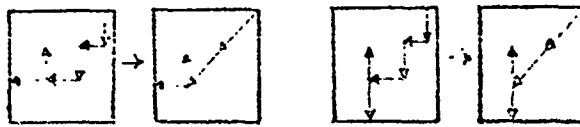
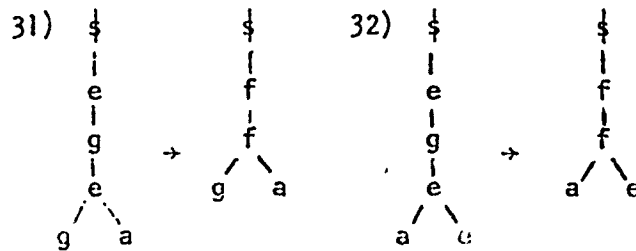
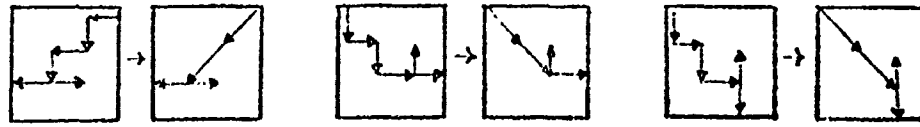


The graphical interpretation of the transformational grammar is given under each rule in terms of non-smooth and smoothed patterns.









4.1.2 Texture Region Primitive Extraction

The texture region primitive extraction takes the digitized image as input and perform texture analysis to obtain the texture region primitives. This process consists of the sub-processes histogram equalization, variability texture measurement, and texture region primitive assignment.

1) Histogram Equalization: The image of interest, $P(x,y)$, is stored as an $M \times M$ array in the computer memory. Each element of this set has a grey level value given by the intensity function $z(x,y)$. A histogram of an image is defined as follows [83]: Given an image f , let $P_f(z)$ denote the relative frequency with which grey level z occurs in f , for all z in the grey level range $[z_1, z_k]$ of f . The graph of $P_f(z)$ as a function of z , normalized so that $\int_{z_1}^{z_k} P_f(z) dz$ is equal to the area of f is called the histogram of f . The histogram equalization technique [72] requantizes the grey levels to k' levels which have $z_1, z_2, \dots, z_{k'}$ as the values of the requantization interval points, with the values of

z_1 and z_k equal to the minimum and maximum values of z_i respectively. The histogram equalization technique reassigns values of grey levels to the picture pixels. This technique assigns an equal number of picture pixels to each interval of the grey level z_{k1} . Thus, the number of grey levels of the original digitized image (usually 256 or 128 grey levels) is reduced to a reasonably smaller number (e.g., 16 or 8 grey levels). Even if, for example, two images of the same scene had different digitized values due to the fact that they were digitized on different machines under different amounts of brightness, the histogram equalization technique would assign values to pixels according to their relative values to each other, and, thus, the histogram equalization results of these two images of the same scene would be the same.

2) Variability Texture Measurement. Texture information is extracted from the spatial relationship of the grey levels of pixels in the image. The joint probability density of the pairs of grey levels that occur at pairs of points with distance d is calculated. If there are k grey levels, the array is a $k \times k$ matrix $P(i,j)$, called a co-occurrence matrix [73]. Then a variability texture feature is calculated to measure the spatial relationship of the grey levels of an image. Havalick [72] suggested twelve texture feature measurements for terrain classification. Here, we use a method of texture analysis for image segmentation. The variability texture feature is the modified entropy texture feature [72]. (entropy $= -\sum_i \sum_j \left(\frac{P(i,j)}{R} \right) \log \left(\frac{P(i,j)}{R} \right)$). The logarithm portion $\left(\log \left(\frac{P(i,j)}{R} \right) \right)$ of the entropy texture feature is modified to $\log \left(\frac{P(i,j)}{R} \right)^K$. R is the normalization constant of the matrix $P(i,j)$. K is the range factor to expand the range of the values

of the texture measurements. The variability texture is thus defined as [72]:

$$\text{VARIABILITY} = - \sum_i \sum_j \left(\frac{P(i,j)}{R} \right)^K \log \left(\frac{P(i,j)}{R} \right), \quad K \geq 1$$

In the previous research work in texture analysis for terrain classification [72,73], there is no indication which texture feature is the best for terrain classification. Therefore, the variability texture feature and some other texture features such as second order moment, contrast, and correlation are measured on the same image [72]. From our experiments on images of several areas in Indiana, the variability texture measurement characterizes the major land-use classes as agricultural, wooded, old residential, new residential, and watery areas. Thus, the variability texture measurement is used for image segmentation. The use of more than one texture feature (in addition to variability texture) could be useful to image segmentation. The reason that only one texture feature is used, is that the computer processing time increases as more texture features are calculated. The preprocessed and primitive extraction result is to be processed by syntactic analysis. Hence, one texture feature, variability, is used in the preprocessing and primitive extraction. Figure 4.2 is an illustration of the variability texture measurements of an 11 x 11 image window. The distance is one for the horizontal, vertical, left diagonal, and right diagonal co-occurrence matrices. The variability texture is calculated on each of the four matrices, separately. The average value of these four measurements is taken as the texture measurement of the center (4x4 pixels) of that image window.

```

888888888888
888888888888
866888888788
888888886453
587888886444
18755441311
38744431111
35883532111
22484311211
25258822211
34445855552

```

CO-OCCURRENCE MATRIX PH

18	4	4	1	0	0	0	1
4	6	1	1	4	0	0	1
4	1	0	3	4	0	0	2
1	1	3	14	3	2	1	2
0	4	4	3	8	0	1	5
0	0	0	2	0	2	0	4
0	0	0	1	1	0	0	6
1	1	2	2	5	4	5	72

CO-OCCURRENCE MATRIX PV

20	6	3	2	2	1	0	0
6	4	2	2	5	0	0	0
3	2	4	5	1	0	0	2
2	2	5	4	6	0	1	5
2	5	1	6	0	0	0	7
1	0	0	0	0	2	0	5
0	0	0	0	1	0	4	3
0	0	2	5	7	5	3	66

CO-OCCURRENCE MATRIX PLD

16	4	2	3	1	0	0	2
4	4	3	2	4	0	0	0
2	3	2	3	1	1	0	1
3	2	3	2	7	1	1	3
1	0	1	1	2	0	0	3
0	0	0	1	0	0	0	5
0	0	0	1	2	0	0	5
2	0	1	5	3	6	5	60

CO-OCCURRENCE MATRIX PRD

14	5	3	2	2	0	0	2
5	4	0	1	3	0	0	3
3	0	2	3	2	0	0	4
2	1	3	4	6	0	0	5
2	3	2	6	0	0	0	5
0	0	0	2	0	0	1	5
0	0	0	0	0	0	0	5
2	3	4	6	5	5	5	52

VARIABILITY TEXTURE PH = 79.0509

VARIABILITY TEXTURE PV = 81.1867

VARIABILITY TEXTURE PLD = 82.5939

VARIABILITY TEXTURE PRD = 85.6009

VARIABILITY TEXTURE MEASUREMENT = 82.1090

Figure 4.2. An illustrative example of variability texture measurement on an 11x11 image window.

In segmenting a large image, the window size used for texture measurement is 11×11 pixels. The variability texture feature is measured from the 11×11 window and the value of this measurement is the texture value of the center 4×4 cell. This technique was devised because it is quite possible for a textural area to be smaller than the window size, or that the boundary between different textural areas lies in the operation window. In this proposed technique, such a problem is taken care of as we locate the boundaries by shifting the 11×11 operation window 4 pixels at a time. Thus, the potential boundaries could be preserved and the spatial relationship can still be extracted because the window size (11×11) has not been reduced. Several other sizes of operation window such as 9×9 , 10×10 , 12×12 and 13×13 were also tried. The texture measurements from the smaller windows sometimes failed to give the same good results as did those from the 11×11 window. The reason for this is that the measurements from smaller window did not yield enough global information of the image data to allow the extraction of the proper texture measurements. The results from the texture measurements of larger windows were similar to those of the 11×11 window. The comparative computational time for these measurements by different windows is that the time for a 9×9 window is about 5% less than that for the 11×11 , and the time for the 13×13 is about 10% more. From this study of texture measurements with respect to window sizes, the 11×11 window was used for the texture measurements in our experiments of the LANDSAT images.

3) Texture Region Primitive Assignment. After obtaining the texture values for 4×4 unit cells, the histogram of the texture values in the texture domain is thresholded and then assign texture codes to the

segments. The histogram is made by shifting the 11x11 window 11 pixels at a time, because only the global distributions of the texture values are of interest here. The feasibility of variability texture measurements and this technique can be shown as follows: Figure 4.3 is the histogram result of variability texture measurements on a 385x198 pixels LANDSAT image of an Indiana area. Figure 4.4 is the histogram result of angular second moment (Angular second moment = $\sum_i \sum_j \{P(i,j)\}^2$). Figure 4.5 is the histogram result of texture measurements of contrast (Contrast = $\sum_{n=0}^{Ng-1} n^2 \{ \sum_{i=1}^{Ng} \sum_{j=1}^{Ng} P(i,j) \}$). Figure 4.6 is the histogram result of texture

measurements of correlation. (Correlation = $(\sum_i \sum_j (i \cdot j) P(i,j) - \mu_x \mu_y) / \sigma_x \sigma_y$, μ_x , μ_y , σ_x and σ_y are the means and standard deviations of P_x and P_y). From the histogram of the texture measurements, the 4x4 cells are grouped to different texture regions.

4.1.3 Boundary Primitive Extraction

Following the texture region primitive extraction is the boundary primitive extraction consisting of horizontal processing, vertical processing, logic integration, and syntactic line smoothing.

1) Horizontal Processing. The Horizontal Processing processes the "texture region primitive extracted image" row-wise to locate the potential horizontal boundary segments. The operation procedure is as follows: let $Q(I,J)$ be the picture function at location (I,J) .

Step 1. Start with $Q(I,J)$ as reference.

Step 2. Compare $Q(I,J)$ with $Q(I,J+1)$. If the distance is smaller than a specified value, a "zero" is set on $Q(I,J)$ and $Q(I,J+1)$. Then $Q(I,J)$ and $Q(I,J+2)$ are compared. If the



Figure 4.3. Histogram of variability texture measurements.



Figure 4.4. Histogram of angular second moment texture measurements.

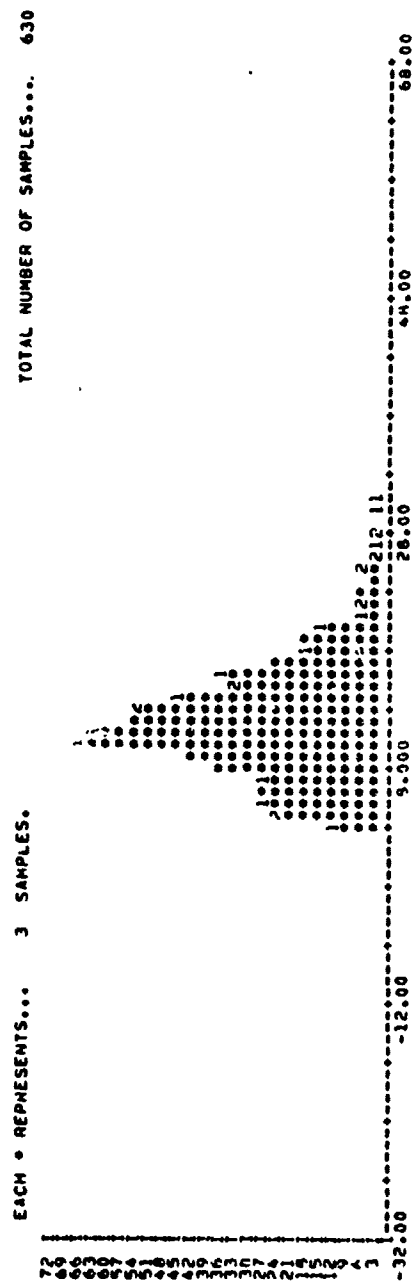


Figure 4.5. Histogram of contrast texture measurements.



Figure 4.6. Histogram of correlation texture measurements.

distance is greater than or equal to the specified value.

A "one" is set for $Q(1, J+2)$ as a potential boundary primitive. Then the same process is applied with $Q(1, J+3)$ as the reference.

Step 3. When this process is operated to the rightmost of the row, the $Q(1+1, J)$ is the reference and Step 2 is applied until all the rows are processed.

The idea of this process is to treat the image matrix as independent rows. After this process the potential vertical boundaries of the image are detected. The reason for comparing $Q(1, J)$ and $Q(1, J+2)$ (when $Q(1, J+1)=0$) in Step 2, is because the reference must be kept in the same operation. If instead of comparing $Q(1, J)$ and $Q(1, J+2)$, $Q(1, J+1)$ and $Q(1, J+2)$ are compared. The reference is shifted, thus none of the boundary primitives will be detected.

2) Vertical Processing. The Vertical Processing is similar to the Horizontal once except that it processes the image column-wise to locate the potential horizontal boundary segments.

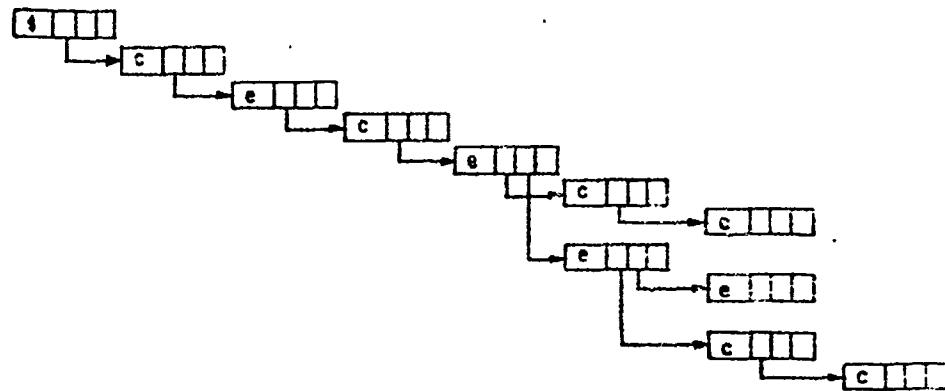
3) Logic Integration. The result of horizontal processing is defined as H and the result of vertical processing is defined as V . The Logic Integration is a boolean "OR" function of H and V and it is defined as $R(H, V)$.

H	V	$R(H, V)$
0	0	0
0	1	1
1	0	1
1	1	1

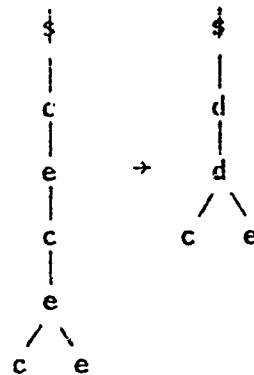
The software program is implemented to compare $H(I,J)$ and $V(I,J)$ to obtain the logic integration result.

After the process of logic integration, the potential boundaries of the picture are detected. The deformation of the boundaries is to be removed by the line smoothing technique which is a syntactic process of transformational grammar.

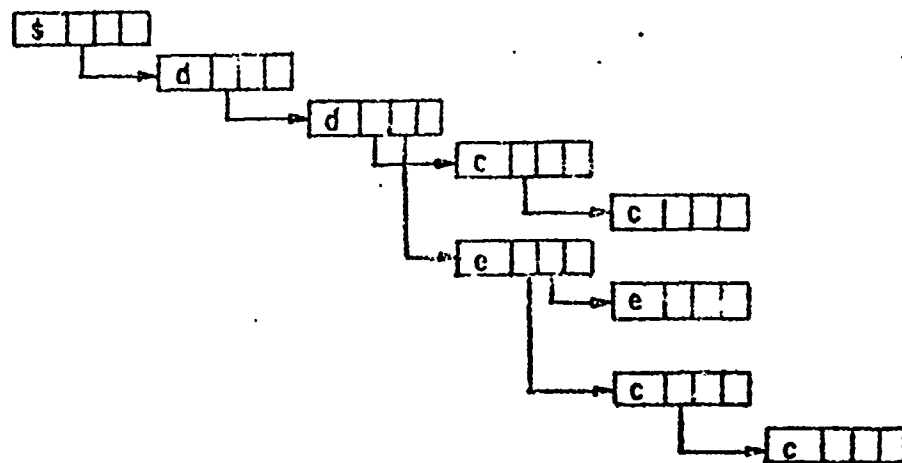
4) Syntactic Line Smoothing. A tree transformational grammar is designed to reduce irregularities and smooth the patterns. The set of tree transformational grammar for syntactic line smoothing is inferred by the inference scheme in section 4.1.1. In implementing the tree transformation, first, the image window is encoded into the tree language which describe the boundaries in that window. The output of the logic integration is the binary image window. The encoding procedure checks the window row by row to find the first pixel with non-zero value as the starting point (\dagger). Then, a search for surrounding non-zero points is followed. The non-zero points are encoded as the primitives. These primitives are the 2x2 blocks, a \uparrow , b \nearrow , c \rightarrow , d \searrow , e \downarrow , f \nwarrow , g \leftarrow , and h \uparrow which were graphically shown in section 4.1.1. If only one branch is found from the starting point, then a pointer is used to link the connecting primitive of this branch with the starting primitive. If two branches are found, then two pointers are used to link the encoded primitive of each branch respectively. The same encoding procedure is applied to each branch to obtain the tree structure of the primitives in the image window. This tree structure is thus the tree language for the tree grammar analysis. For example, for an input window from logic integration process as



In processing the tree representation of the input pattern by the tree transformation grammar, the rule (1)



will produce the transformed pattern and its data representation is as follows.



In order to apply the tree transformation grammar to smooth the pattern, the pattern primitives around the pattern primitive of interest have to be checked. In a 6x6 window, there are eight primitive windows (2x2) around the center primitive window of interest. Thus, there is enough information for deciding whether the proper smoothing rule will be applied to smooth the pattern or not. Therefore, in processing an image, the syntactic line smoothing process is performed on a 6x6 window. The window is shifted 4 pixels at a time to repeat the process. There are always 2 pixels overlapping on each process, thus the irregularities between two neighboring windows will also be processed by this line smoothing technique. When the window operation reaches the edge of the image, it moves downward four pixels and starts from the leftmost point of the image to repeat the same process, until the whole image is processed by this syntactic line smoothing technique.

4.1.4 Tree Grammar Analysis

The recognition by tree grammar analysis is performed by the tree automaton corresponding to the tree grammar. The result of boundary primitive extraction is encoded window by window as tree structure by the encoding procedure described in section 4.1.3. If the input tree structure can be derived by the transition rules of tree automaton to the final states, then this tree is accepted by the tree automaton. If the input tree structure cannot be derived to the final states by the tree automaton, then the input structure is rejected.

By continually performing this procedure on each image window, the boundary structures of the image are analyzed and the syntactically correct boundaries of image segments are obtained. Hereby, the image segmentation result is achieved.

Tree Grammar Analysis Algorithm

Input: Boundary Primitive Extraction Result, Binary Image, $P(I,J)$.

Output: Image Segmentation Result.

Algorithm:

1. Construct the tree automaton from the inferred tree grammar of the boundary structures. The procedure of tree automaton construction has been described in section 4.1.1.
2. Initialize the array $P(I,J)$ as the operation window (8×8). Load the data of image segment $I(I,J)$ where $J = 1,8$ and $I = 1,8$ onto the operation window $P(I,J)$.
3. Encode the array $P(I,J)$ into a tree structure as described in the section on syntactic line smoothing.
4. Each transition rule of tree automaton is stored in computer as a linear array. For example, $f_{\frac{1}{2}}(q_{-1}) = q_s$ is stored as a 1×3 array called $TA(I,J)$, then $TA(1,1) = q_s$ (present state), $TA(1,2) = \frac{1}{2}$ (input state), and $TA(1,3) = q_{-1}$ (next state). The input is stored as a linked tree data structure. The root ($\frac{1}{2}$) of the tree is the input symbol and the initial state (present state) is q_s , thus, a next state is obtained by searching for the transition rule, from the array $TA(I,J)$, with the present state q_s and input symbol $\frac{1}{2}$. The recognition of each branch of the input tree is similar to that of a finite state automaton. If all the branches of the input tree can achieve the final states by applying the transition rules of the tree automaton, then the input tree is accepted. If there is one or more branches which cannot achieve the final branches which cannot achieve the final states by

checking all the transition rules of the tree automaton, then this input tree is rejected. The syntactically correct boundary patterns are thus accepted as the image segmentation result.

5. Shift four columns to the right of $M(I,J)$. Then go to Step 3 until the shifting operation reaches the rightmost column. Go to Step 6.
6. Shift four rows down, go to Step 3 until reaching the last row of the image $M(I,J)$. Then the image $M(I,J)$ is analyzed by the tree automaton and the syntactically correct boundaries are accepted as the image segmentation result. The flow chart of the algorithm is given in Appendix H.

For example, using the grammar inferred in the example of grammatical inference in section 4.1.1, the tree grammar G_t is as follows:

$$G_t = (V, r, P, S)$$

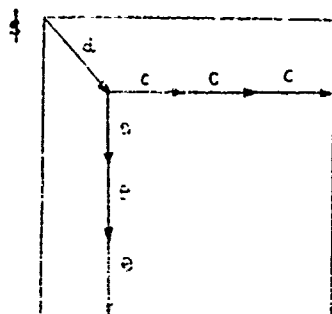
$$V = \{A_1, A_2, S, \frac{1}{2}, c, d, e, f\}$$

$$V_T = \{\frac{1}{2}, \rightarrow c, \rightarrow d, \rightarrow e, \rightarrow f, \}$$

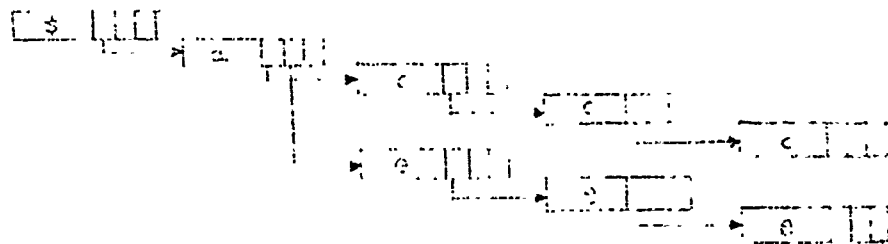
$$r(s) = \{1\}, r(c) = \{0,1,2\}, r(d) = \{2\}, r(e) = \{0,1\}, r(f) = \{1,1\}$$

$$\begin{array}{ll}
 P: s \rightarrow \begin{array}{c} \frac{1}{2} \\ | \\ A_1 \end{array} & s \rightarrow \begin{array}{c} \frac{1}{2} \\ | \\ A_2 \end{array} \\
 \\
 A_1 \rightarrow \begin{array}{c} c \\ / \quad \backslash \\ A_1 \quad A_2 \end{array} & A_2 \rightarrow \begin{array}{c} d \\ / \quad \backslash \\ A_1 \quad A_2 \end{array} \\
 \\
 A_1 \rightarrow \begin{array}{c} c \\ | \\ A_1 \end{array} & A_2 \rightarrow e \\
 \\
 A_2 \rightarrow \begin{array}{c} e \\ | \\ A_2 \end{array} & A_1 \rightarrow c
 \end{array}$$

The input pattern is encoded as



The input tree structure is



As described in section 4.1.1, a tree automaton can be constructed corresponding to the tree grammar of this example. The tree automaton is thus constructed as M_t , $M_t = (Q, f_c, f_e, f_d, f_{\frac{1}{d}}, F)$, where $Q = \{q_c, q_e, q_1, q_2, q_s\}$, $F = \{q_c, q_e\}$, and f : (The detailed steps of tree automaton construction are given in Appendix I).

- (1) $f_{\frac{1}{d}}(q_1) = q_s$
- (2) $f_{\frac{1}{d}}(q_2) = q_s$
- (3) $f_c(q_1, q_2) = q_1$
- (4) $f_d(q_1, q_2) = q_2$
- (5) $f_c(q_1) = q_1$
- (6) $f_e(q_2) = q_2$
- (7) $f_c(q_c) = q_1$
- (8) $f_e(q_e) = q_2$

After constructing the tree automaton, the recognition of an input tree can be performed by the tree automaton. The recognition of the input tree of this example is as follows: the root of the input tree is $\frac{1}{2}$. The first transition rule of the automaton is applied to obtain the next state q_1 . The second level of the input tree is d . The transition rule, which has d as input and q_1 as present state, cannot be found. Thus, the second transition rule should be used to obtain the next state q_2 in the recognition of the first level of the input tree. The second level of the input tree is d . The transition rule with present state q_2 and input d , which is transition rule (4), produces the next state pair (q_1, q_2) . The third level of the input tree is branches c and e . With present state q_1 and input symbol c , the transition rule (5) produces the next state q_1 . The fourth level of the input tree is c . The transition rule (5) is applied again to produce the next state q_1 . The final level of the input tree is c . The transition rule (7) is applied to achieve the final state q_c . The branch, d , of the third level of the input tree is recognized by transition rules (6), (6), and (8) for the fourth, fifth and final level of the input tree to achieve the final state. Thus, all the branches of the input tree achieve final states, then the tree is accepted by the tree automaton.

4.2 COMPUTER EXPERIMENTAL RESULTS OF IMAGE SEGMENTATION FROM LANDSAT IMAGES

The proposed syntactic method for image segmentation was implemented on the IBM 360/67 computer at the Laboratory for Applications of Remote Sensing (LARS). The experiments were conducted on various LANDSAT and infrared images.

Applying the grammatical inference procedure of tree grammar in section 4.1.1, a tree grammar was inferred to describe the boundaries of image segments. This grammar, as stated in section 4.1.1, was applied to the image segmentation of LANDSAT images. The tree automaton corresponding to the tree grammar is constructed as M_t . $M_t = (Q, f_a, f_b, f_c, f_d, f_e, f_f, f_g, f_h, f_i, F)$. Where

$$Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_a, q_b, q_c, q_d, q_e, q_f, q_g, q_h, q_s\}$$

$$F = \{q_a, q_b, q_c, q_d, q_e, q_f, q_g, q_h\}$$

$$f_i(q_1, q_2) = q_s$$

$$f_i(q_1) = q_s$$

$$f_i(q_6) = q_s$$

$$f_i(q_2) = q_s$$

$$f_i(q_7) = q_s$$

$$f_e(q_1, q_2) = q_1$$

$$f_c(q_1, q_2) = q_2$$

$$f_c(q_1, q_2) = q_1$$

$$f_d(q_1, q_2) = q_2$$

$$f_f(q_1, q_2) = q_1$$

$$f_g(q_1, q_2) = q_2$$

$$f_d(q_3, q_4, q_5) = q_1$$

$$f_e(q_3, q_4, q_5) = q_1$$

$$f_f(q_1, q_2) = q_2$$

$$f_d(q_3) = q_3$$

$$f_e(q_3) = q_3$$

$$f_f(q_3) = q_3$$

$$f_g(q_3) = q_3$$

$$f_d(q_4) = q_4$$

$$f_e(q_4) = q_4$$

$$f_a(q_4) = q_4$$

$$f_c(q_3) = q_3$$

$$f_h(q_3) = q_3$$

$$f_b(q_4) = q_4$$

$$f_f(q_4) = q_4$$

$$f_g(q_4) = q_4$$

$$f_c(q_4) = q_4$$

$$f_h(q_4) = q_4$$

$$f_b(q_5) = q_5$$

$$f_d(q_5) = q_5$$

$$f_e(q_5) = q_5$$

$$f_e(q_4) = q_1$$

$$f_f(q_5) = q_5$$

$$f_g(q_5) = q_5$$

$$f_c(q_5) = q_5$$

$$f_h(q_5) = q_5$$

$$f_e(q_5) = q_5$$

$$f_e(q_1) = q_6$$

$$f_c(q_7) = q_7$$

$$f_c(q_1) = q_7$$

$$f_e(q_1) = q_1$$

$$f_c(q_1) = q_1$$

$f_f(q_1) = q_1$	$f_b(q_2) = q_2$	$f_c(q_2) = q_2$
$f_d(q_2) = q_2$	$f_g(q_2) = q_2$	$f_f(q_2) = q_2$
$f_b(q_3) = q_3$	$f_c(q_c) = q_1$	$f_f(q_f) = q_1$
$f_b(q_b) = q_2$	$f_c(q_c) = q_2$	$f_d(q_d) = q_2$
$f_g(q_g) = q_2$	$f_f(q_f) = q_2$	$f_b(q_b) = q_3$
$f_d(q_d) = q_3$	$f_e(q_e) = q_3$	$f_f(q_f) = q_3$
$f_g(q_g) = q_4$	$f_d(q_d) = q_4$	$f_e(q_e) = q_4$
$f_f(q_f) = q_4$	$f_g(q_g) = q_4$	$f_c(q_c) = q_4$
$f_h(q_h) = q_4$	$f_b(q_b) = q_5$	$f_d(q_d) = q_5$
$f_e(q_e) = q_5$	$f_f(q_f) = q_5$	$f_g(q_g) = q_5$
$f_c(q_c) = q_5$	$f_h(q_h) = q_5$	$f_e(q_e) = q_6$
$f_c(q_c) = q_7$	$f_a(q_a) = q_4$	$f_e(q_e) = q_1$
$f_b(q_b) = q_4$	$f_c(q_c) = q_3$	$f_h(q_h) = q_3$

Figure 4.7 is a LANDSAT image of Bloomington, Indiana (88x88 in size). Figure 4.8(a) is the intermediate result after the texture region primitive extraction of the proposed image segmentation method. The syntactic image segmentation algorithm achieved the segmentation of the picture in Figure 4.7. The result is shown in Figure 4.8(b). For the purpose of knowing the ground truth of this satellite image, this area has been classified by a maximum-likelihood point-by-point classifier. This classification result is shown in Figure 4.9 in which A stands for agricultural, T for forest, X for old residential, Y for new residential, and W for watery areas. The computer processing time of the syntactic method is only 55 seconds. But the classification technique takes 240 seconds of CPU time.

Consider now the performance of the syntactic image segmentation algorithm. For example, the left upper corner (line 1-4, columns 12-24) of Figure 4.9, the ground truth of the image Figure 4.7, is a small area of forest area. This area is well segmented by the proposed method as a homogeneous region shown in the corresponding coordinates in Figure 4.8(b). One good thing about this algorithm is that in the segmentation result, the boundaries of the image are always closed. A closed boundary of image segments contributes to the future recognition of either the shape or the characteristic pattern of that segment.

4.3 COMPUTER EXPERIMENTAL RESULTS OF IMAGE SEGMENTATION FROM FLIR (FORWARD LOOKING INFRARED) IMAGES

The first step of an object detection problem is actually an image segmentation problem. The object of interest must be first extracted from the scene. In the human visual system, the object is detected by the human eye through the characteristic contents and/or the shape of the object. The syntactic image segmentation algorithm described in section 4.1 tends to simulate the human visual system for object detection. The texture analysis of the preprocessing extracts the characteristic contents of the image. The syntactic analysis examines the boundary of the object. The syntactic image segmentation algorithm is also adaptable to object detection from FLIR images. When the image is rich in texture, the syntactic method for image segmentation described in section 4.1 can be applied to object detection. When the image is not rich in texture, the texture feature measurement of the preprocessing part of the syntactic method is changed to a mean vector measurement. The syntactic analysis is unchanged.



Figure 4.7. Satellite image of Indiana area.

TEXTURE REGION

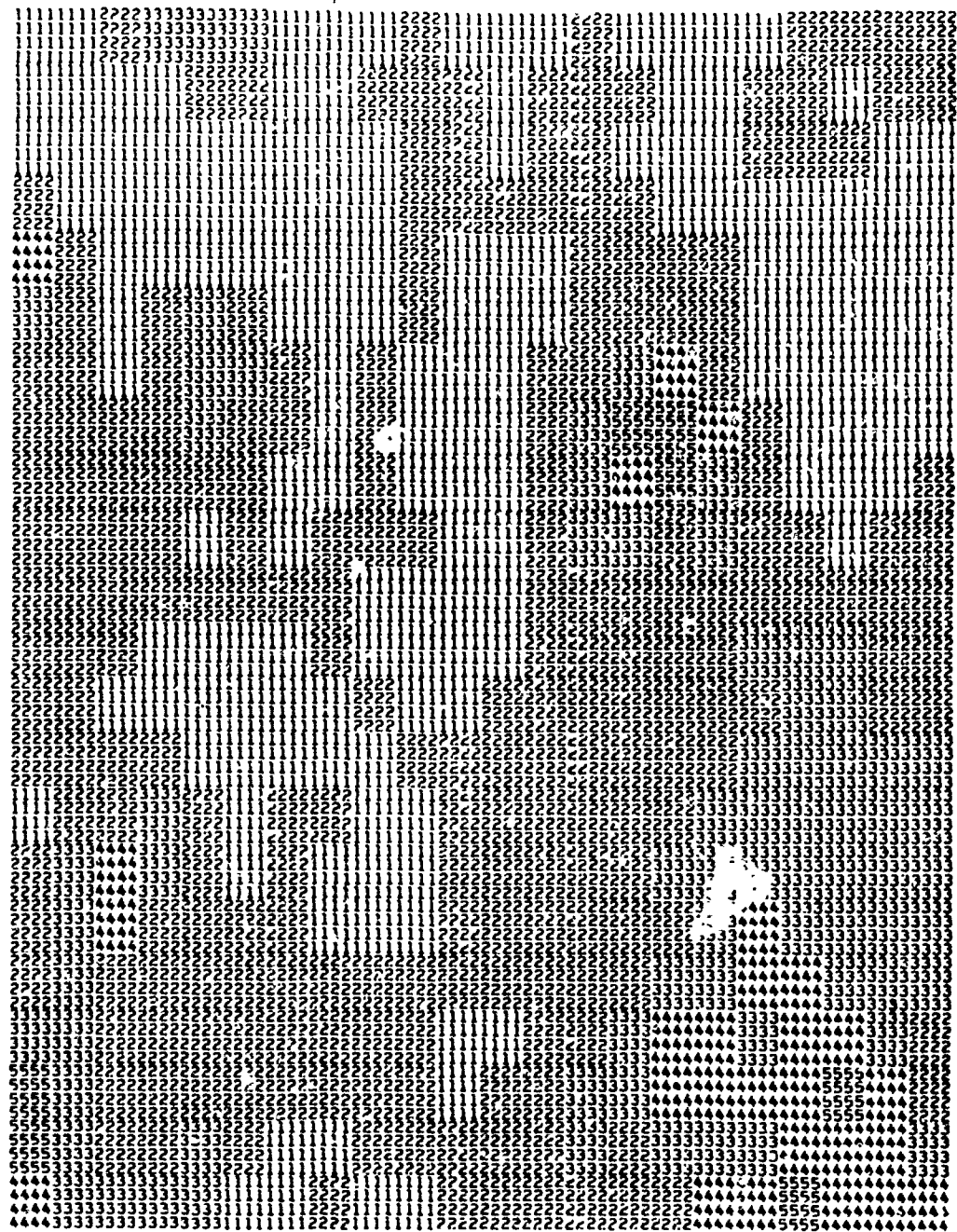


Figure 4.8(a). Intermediate result after the texture region primitive extraction of the syntactic image segmentation algorithm on Figure 4.7.

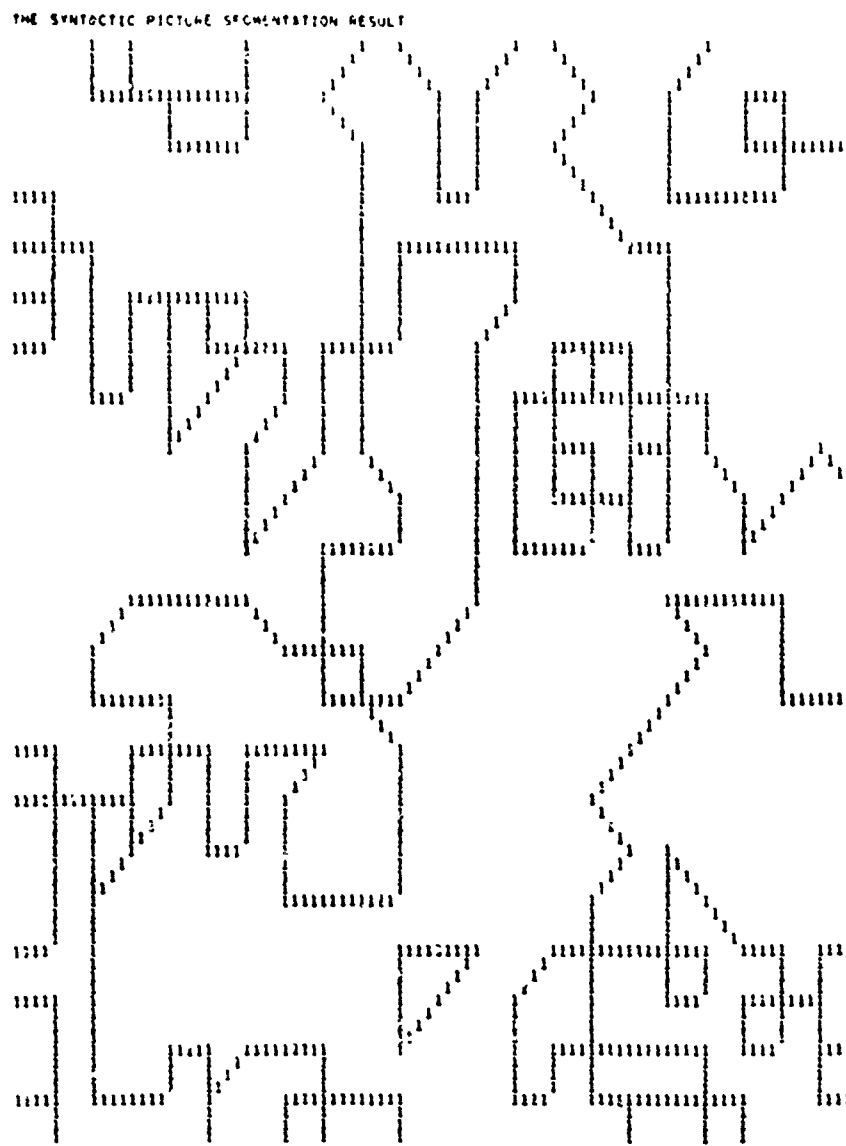


Figure 4.8(b) Result of syntactic image segmentation of Figure 4.7.

Figure 4.9. Point-by-point classification result of Figure 4.7.

If several objects have the same texture contents, but different shapes, then the object of interest is primarily distinguished from the other objects by the second process. In this case the first process contributes to the extraction of those objects from their background. The syntactic analysis process examines the boundaries of these objects and accepts those objects with the boundaries which can be generated by the grammar.

If several objects have the same shapes, but different texture contents, the first process in the syntactic method extracts the texture contents of the objects. If the contents are the same as those of the object of interest, then the syntactic analysis examines the boundary of the object. If the syntactic analysis has a successful parsing, then the object of interest is still distinguished from other objects in the image scene.

This method can also be applied to tactical target detection from infrared images of battleground scenes. The technique used here is the same as that which has been described in section 4.1 basically.

4.3.1 Data Acquisition System of Infrared Images

The infrared images were obtained from Honeywell Systems and Research Center [118] under a consulting contract. The Honeywell Corporation obtained the FLIR imagery using Navy P2-V aircraft at the Naval Air Test Center in Patuxent River, Maryland, and a Honeywell 18 detector serial scan FLIR sensor. The P2-V is outfitted as an electro-optical test bed allowing the collection and recording of taped imagery of various kinds. The infrared images were obtained in June of 1974 and were of military vehicles at Camp A P Hill in southern Maryland. Altitudes during the flight were about 3,000 feet.

The proposed technique of image segmentation has also been tested on a different data set of infrared images which were collected by the Night Vision Laboratory. Figure 4.10(a), 4.13(a), 4.14(a) and 4.15(a) are the test images from Honeywell and Figures 4.11(a), 4.12(a), 4.16(a), 4.17(a), and 4.18(a) are the test images from the Night Vision Laboratory.

4.3.2 Experimental Results

Since infrared images are thermal images, the characteristics of these images are different than those of the LANDSAT images. Also, the objects of interest in the object detection from infrared images are tactical targets, larger ones and small ones, such as military vehicles. Basically, the algorithm for the experiments on infrared images is the same as the algorithm described in section 4.1. The differences are the window size for the texture measurements and a different tree grammar to describe the boundaries of military vehicles. Experiments on different window sizes for texture measurements were carried out. A window size of 8×8 provided the best segmentation result in the data set used. Smaller window sizes tended to give segmentation results of the local property, and larger windows took more CPU time while providing no better results. Therefore, the window size used for texture measurements in our experiments on infrared images was 8×8 .

The texture region primitive extraction and boundary primitive extraction were the same here as those in section 4.1. The tree grammar for describing the boundary structures of military vehicles is inferred by applying the tree grammar inference procedure in section 4.1.1 to the sample patterns of the boundary structures of military vehicles (Appendix F). The resultant tree grammar is G_t :

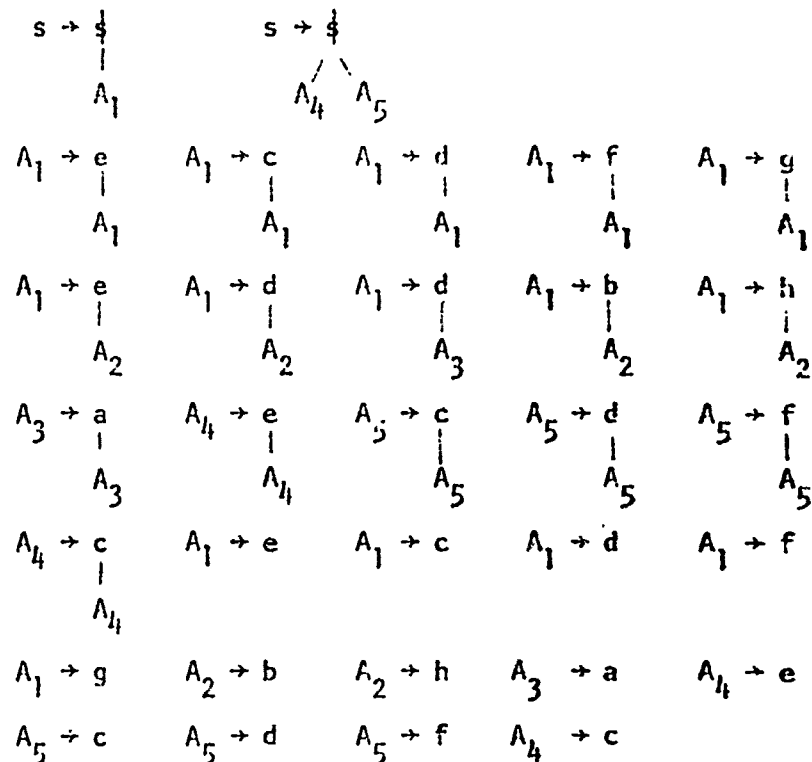
$$G_t = (V, r, P, S)$$

where

$$V = \{S, \$, A_1, A_2, A_3, A_4, A_5, a, b, c, d, e, f, g, h\}$$

$$r(\$) = \{2, 1, 0\}, r(a) = \{1, 0\}, r(b) = \{1, 0\}, r(c) = \{1, 0\}, \\ r(d) = \{1, 0\}, r(e) = \{1, 0\}, r(f) = \{1, 0\}, r(g) = \{1, 0\}, r(h) = \{1, 0\}$$

rule P:



Since the shapes of the objects of interest, military vehicles, in the image segmentation of FLIR images are simple, most of the grammar rules happen to be string (regular) grammar rules.

The tree automaton corresponding to the tree grammar G_t is M_t .

$$M_t = (Q, f_a, f_b, f_c, f_d, f_e, f_f, f_g, f_h, f_\$, F)$$

where

$$Q = \{q_1, q_2, q_3, q_4, q_5, q_a, q_b, q_c, q_d, q_e, q_f, q_g, q_h, q_s\}$$

$$F = \{q_a, q_b, q_c, q_d, q_e, q_f, q_g, q_h\}$$

$$f_\$(q_1) = q_5 \quad f_f(q_5) = q_5$$

$$\begin{array}{ll}
f_{\frac{1}{2}}(q_4, q_5) = q_5 & f_c(q_4) = q_4 \\
f_e(q_1) = q_1 & f_e(q_e) = q_1 \\
f_c(q_1) = q_1 & f_c(q_c) = q_1 \\
f_d(q_1) = q_1 & f_d(q_d) = q_1 \\
f_f(q_1) = q_1 & f_f(q_f) = q_1 \\
f_g(q_1) = q_1 & f_g(q_g) = q_1 \\
f_e(q_2) = q_1 & f_b(q_h) = q_2 \\
f_d(q_2) = q_1 & f_h(q_h) = q_2 \\
f_d(q_3) = q_1 & f_a(q_a) = q_3 \\
f_b(q_2) = q_2 & f_e(q_e) = q_4 \\
f_h(q_2) = q_2 & f_c(q_c) = q_5 \\
f_a(q_3) = q_3 & f_d(q_d) = q_5 \\
f_h(q_4) = q_4 & f_f(q_f) = q_5 \\
f_c(q_5) = q_5 & f_c(q_c) = q_4 \\
f_d(q_5) = q_5 &
\end{array}$$

Figure 4.10(a) is an infrared image of a tactical target scene. The size of the image is 88x88. The altitude of the infrared sensor from the ground was about 3,000 feet. The window size used for texture measurements was 8x8. The syntactic analysis was performed by the tree automaton, corresponding the tree grammar for describing the boundaries of military vehicles including tanks, trucks, and armed personnel carriers. The proposed syntactic method successfully segments the infrared image into target and background. The image segmentation result

is shown in Figure 4.10(b). The symbol T represents the boundary of the tactical target. (This image scene is a side view of a truck heading east.) The computation time for image segmentation of Figure 4.10(a) - Figure 4.10(b) is about 50 seconds on the IBM 360167 computer. Several experiments on texture analysis of the distinction between background and targets are conducted. Figure 4.10(c) is a segment of Figure 4.10(a). (Column 21-28, row 61-68 corresponding to the frame 88x88 of Figure 4.10(a)). The texture region primitive measurement result is in Figure 4.10(d) which is a segment of background scene. Figure 4.10(e) is a segment (Column 53-60, row 61-68) of Figure 4.10(a). The texture region primitive measurement result is in Figure 4.10(f). The results of the similar experiments on Figure 4.11(a) are provided from Figure 4.11(c) to Figure 4.11(f).

Figure 4.11(a) is an infrared image which is a side view of a tank. The proposed syntactic method was applied to this image. The tactical target was successfully segmented. The segmentation result of Figure 4.11(a) is given in Figure 4.11(b). The sizes of the images from 4.11(a) to 4.11(b) are all 88x88. Figure 4.12(a) is an infrared image of a vehicle, and the result of its syntactic image segmentation is shown in Figure 4.12(b). Figure 4.13(a) is a top view of a vehicle. The target was successfully segmented even though it is a noisy image of low resolution. The result is in Figure 4.13(b). The image segmentation results from Figures 4.14(a), 4.15(a), and 4.16(a) by the proposed syntactic algorithm are given in Figures 4.14(b), 4.15(b), and 4.16(b) respectively. Experiments involving segmenting small targets were also conducted by

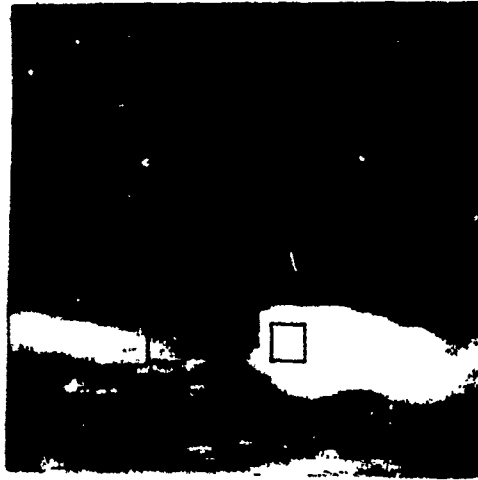


Figure 4.10(a). Infrared image of a tactical target scene.

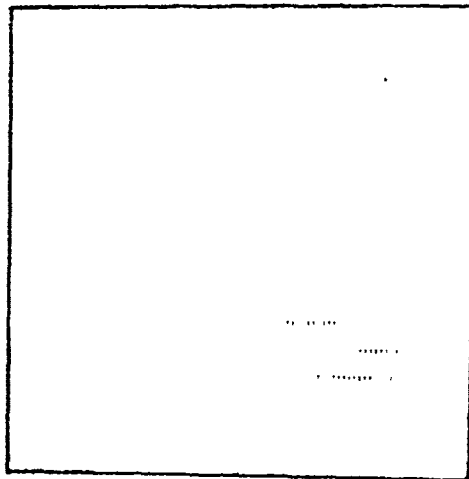


Figure 4.10(b). Image segmentation result by the syntactic method on Figure 4.10(a).



Figure 4.10(c). Image segment
8x8 of the background of Figure
4.10(a).

35	38	38	38	36	36	36	36
36	38	38	38	36	36	36	36
38	38	38	38	36	36	36	36
38	38	38	38	36	36	36	36
34	34	34	34	33	33	33	33
34	34	34	34	33	33	33	33
34	34	34	34	33	33	33	33
34	34	34	34	33	33	33	33

Figure 4.10(d). Texture region
primitive measurement result of
Figure 4.10(c).



Figure 4.10(e). Image segment
8x8 of the target area of Figure
4.10(a).

11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11

Figure 4.10(f). Texture region
primitive measurement result of
Figure 4.10(e).

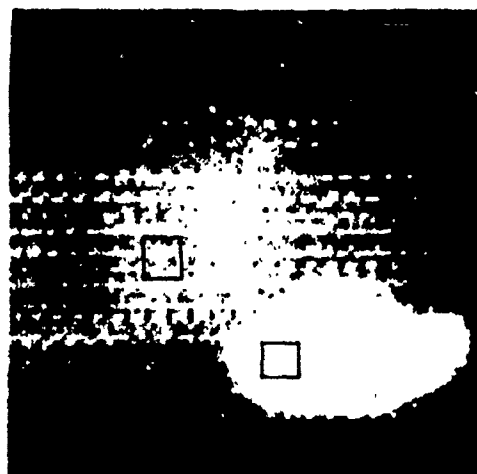


Figure 4.11(a). Infrared image of a tactical target scene.

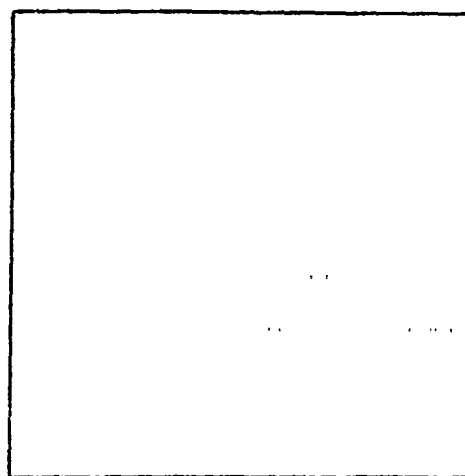


Figure 4.11(b). Image segmentation result by the syntactic method on Figure 4.11(a).



Figure 4.11(c). Image segment
8x8 of the background of Figure
4.11(a).

41	41	41	41	39	39	39	39
41	41	41	41	39	39	39	39
41	41	41	41	39	39	39	39
41	41	41	41	39	39	39	39
39	39	39	39	35	35	35	35
39	39	39	39	35	35	35	35
39	39	39	39	35	35	35	35
39	39	39	39	35	35	35	35

Figure 4.11(d). Texture region
primitive measurement result of
Figure 4.11(e).



Figure 4.11(e). Image segment
8x8 of the target area of Figure
4.11(a).

11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11

Figure 4.11(f). Texture region
primitive measurement result of
Figure 4.11(e).

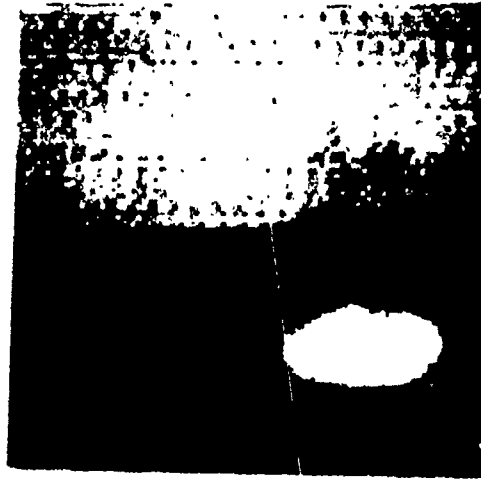


Figure 4.12(a). Infrared image of a tactical target scene.

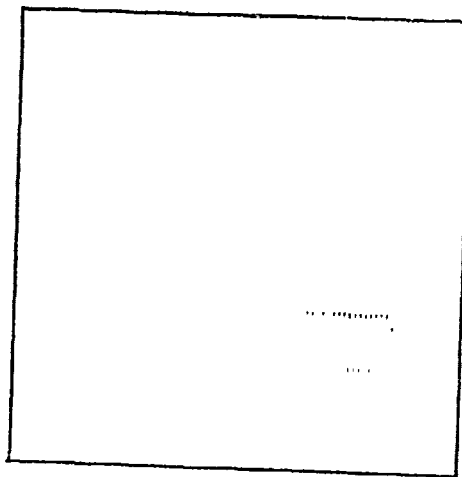


Figure 4.12(b). Image segmentation result by the syntactic method on Figure 4.12(a).



Figure 4.13(a). Infrared image of a tactical target scene.

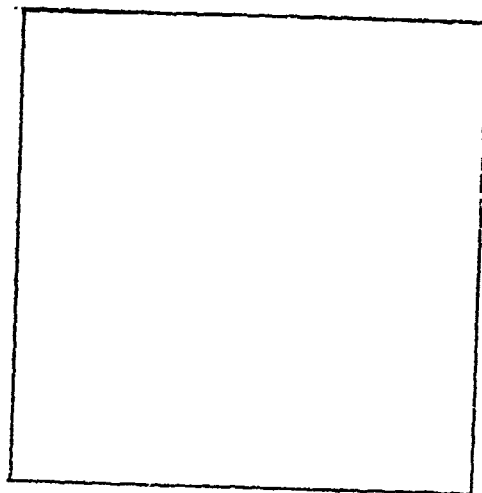


Figure 4.13(b). Image segmentation result by the syntactic method on Figure 4.13(a).

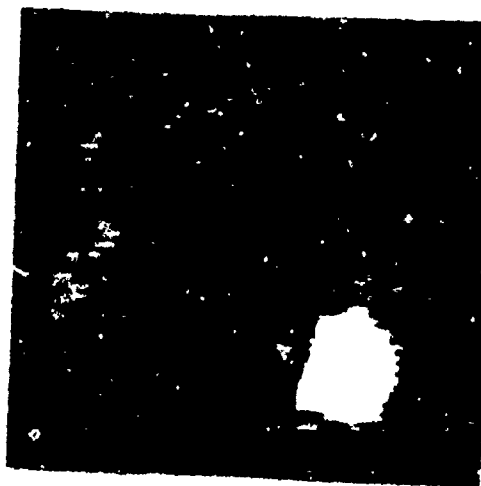


Figure 4.14(a). Infrared image of a tactical target scene.

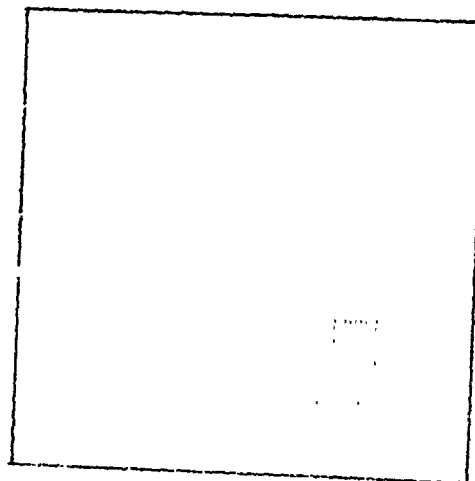


Figure 4.14(b). Image segmentation result by the syntactic method on Figure 4.14(a).

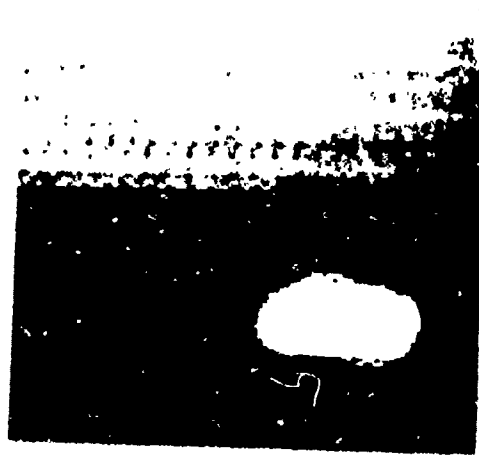


Figure 4.15(a). Infrared image of a tactical target scene.

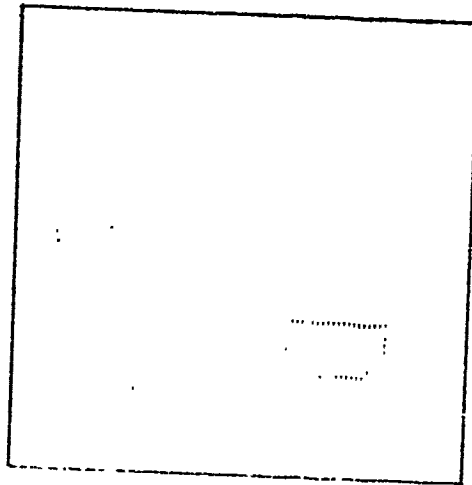


Figure 4.15(b). Image segmentation result by the syntactic method on Figure 4.15(a).



Figure 4.16(a). Infrared image of a tactical target scene.

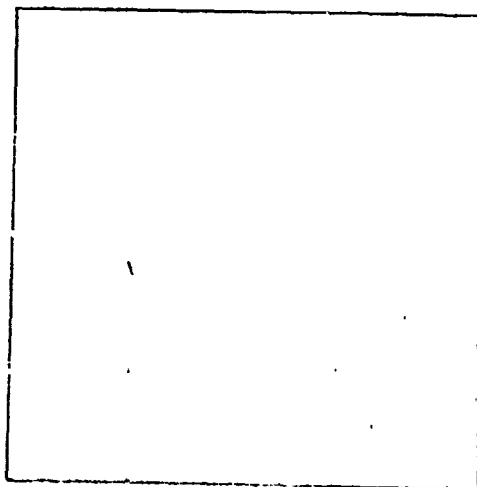


Figure 4.16(b). Image segmentation result by the syntactic method on Figure 4.16(a).



Figure 4.17(a). Infrared image of a tactical target scene.

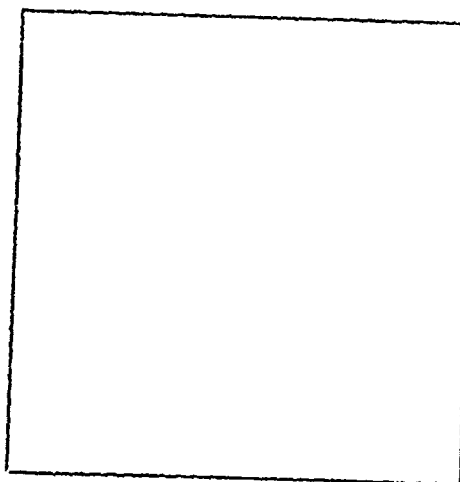


Figure 4.17(b). Image segmentation result by the syntactic method on Figure 4.17(a).

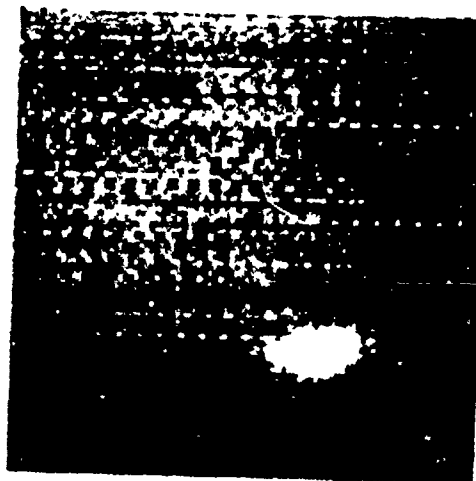


Figure 4.18(a). Infrared image of a tactical target scene.

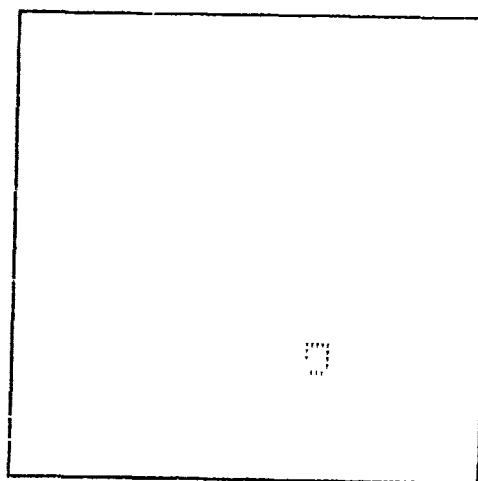


Figure 4.18(b). Image segmentation result by the syntactic method on Figure 4.18(a).

the proposed syntactic algorithm by processing the infrared images of Figures 4.17(a) and 4.18(a). The image segmentation results achieved are shown in Figures 4.17(b) and 4.18(b).

Comparing the segmentation result in Figure 4.10(b) with the test image Figure 4.10(a), the object of interest, a truck, is well segmented. This experiment exemplifies the performance of the proposed method. The boundary contour of the object of interest in Figure 4.10(b) can be used for shape recognition. In the experiment of image segmentation on Figure 4.11(a), a good segmentation result of the test image is also achieved. The objects of interest in Figure 4.10(a) and 4.11(a) are the targets which are larger than most of the targets in Figure 4.12-Figure 4.19. Thus, the image segmentation results have enough boundary information of the targets for the future recognition purpose. Because of the low resolution of the images in Figures 4.12(a), 4.13(a), 4.14(a), 4.15(a), 4.17(a), and 4.18(a), the types of the targets cannot be told even by human. It is hard to distinguish them by the segmentation result either, but those targets are still segmented. The noises usually are caused by heat diffusion of the object and the sensitivity of the infrared sensor and these noises sometime effect the segmentation result. In general, the syntactic image segmentation algorithm still well segments the image into target and background. This contributes to the information extraction from infrared images.

4.4 SUMMARY

This chapter presents a syntactic method for image segmentation. It is a syntax-controlled method which utilizes a tree automaton to extract the boundaries of the homogeneous region segments of an image. The

homogeneity of the segments is measured by texture measurements described in section 4.1.2.

It has been shown from both simulation and experimentation that the proposed syntactic method segments small textural areas as well as larger ones. Thus, this method contributes to the automation of image understanding. As described in section 4.1, the best window size of the texture feature measurement for the proposed method needs to be determined through experiments using different windows on a test data set. Some texture analysis technique, such as syntactic texture modeling and discrimination [117], could be exploited to further refine the texture region primitive extraction of the proposed algorithm.

CHAPTER 5

DESIGN FOR A SPECIAL COMPUTER ARCHITECTURE FOR IMAGE PROCESSING

Image processing by computer encompasses a wide variety of techniques and mathematical tools. In most image processing, large computers have been employed. Unfortunately, the cost is high. As pointed out in the introduction, image processing tasks usually involve an extremely large volume of data, much of which can be operated on in parallel. Therefore, it is important to study special computer architectures for image processing. During the last two decades the field of image processing has grown up rapidly. New techniques, algorithms, and applications have been developed, but there is still a need for improved hardware. A special computer architecture is presented here as a proposal for improving the state-of-the-art in image processing and also to cut costs. Designing this computer architecture was a challenging problem, as the desire was to build a computer that would have the following features:

- 1) The computer was to allow efficient image processing at high speed utilizing interactive computation, making possible large data evaluation.
- 2) The computer was to preserve the general purpose aspects of a general purpose computer for image processing.
- 3) The computer was to be cost effective in order to allow industrial realization.

The framework of this proposed computer architecture consists of a task management processor, a parallel processor, and a sequential arithmetic processor. The task management processor is a set of software system programs serving as an operating system. The parallel processor is proposed because of the parallel nature of the operations involved in image processing. Parallel machines are considered to be particularly suitable for image processing by Thurber and Wald [102]. The parallel processor of the proposed computer architecture consists of an array of microprocessors which allow parallel processing capability. This parallel processor is a homogeneous system in which all processors are alike and are general purpose units. The homogeneous parallel processor lends itself quite readily to extendability as such systems are usually modularly constructed. With modular parallel processing, the system's memory, processor, and input/output modules may be enlarged as processing requirements increase; thereby avoiding replacement of the entire parallel processing system. The modular parallel processor also provides very high reliability since, with several identical modules of each type, the system can withstand failures in several modules and still operate. This arrangement also increases efficiency and through-put since all of the processors could be operating simultaneously. The sequential arithmetic processor is a microprogrammed controlled processor which performs the sequential arithmetics and also controls the input/output devices. The details of these processors will be set forth in section 5.2. It will be shown that the design goals were achieved through the proposed computer architecture.

5.1 PREVIOUS WORK AND COMMENTS

As mentioned in Chapter 1, previous work in the field of special purpose computer architecture for image processing basically falls into two categories: bit-plane processing and distributed processing. The bit-plane processing approach performs the arithmetic computation on the image points which are stored in Boolean bit-plane. For example, if the image has eight grey levels, then the image points are stored in three Boolean planes. The bit-plane processing approach tends to have a large number of processors which perform Boolean operations. The distributed computing approach utilizes processors which have powerful computation capability. These processors are designated by microprogram or hardware to execute certain specific tasks. This configuration forms a distributed computing architecture. In this section we will briefly illustrate each special purpose computer, concentrating on the special features of each computer for image processing.

The Illinois Pattern Recognition Computer, ILLIAC III, [87,100] was designed for automatic scanning and analysis of massive amounts of relatively homogeneous visual data, in particular, bubble chamber negatives. The computation is performed by three units, the pattern articulation unit, the taxicrinc unit, and the arithmetic unit. The pattern articulation unit performs local preprocessing on the digitized raster, such as track thinning, gap filling, line element recognition, etc. The logic desing of the digital computer has been otpimized for the idealization of the input image to a line drawing. Nodes representing end points, bends, points of intersection are labeled in parallel by appropriate programs. The abstract graph describing the interconnection

of labeled nodes is then extracted as a list structure, which comprises the normal output of the processing element (stalactite). The pattern articulation unit consists of an iterative array of 1024 identical stalactites (32×32). A schematic diagram of the stalactite is shown in Figure 5.1. The iterative array of the stalactite can be connected in, either rectangularly or hexagonally, at the programmer's option. Each stalactite can accept an input from any of the eight neighboring stalactites, S_0 , S_1 , ..., and S_7 shown in Figure 5.1 (rectangular) or six neighboring stalactites (hexagonal) in the plane and from itself, M shown in Figure 5.1. The input signals are logically "OR"ed, optionally complemented, and stored in one (or more) of the nine planes. Communication with the supplementary planes in the core buffer is through the "H" plane, which serves as the buffer register of this memory. For output, the output of any selected set of planes can be logically "AND"ed and passed on to neighboring stalactites. With a special signal, the stalactite allows an input signal to pass through it directly, without interim storage. It is this feature which allows path-building within the machine. The set of 32×32 stalactites processes the pixels of a 32×32 pixel image window simultaneously. The whole image is to be processed by one image window (32×32) after another sequentially by the set of 32×32 stalactites. The taxicrinic unit assembles the graphs, which are outputs of the pattern articulation unit, into coherent list structures and categorizes these graphs to complete the visual recognition function. The arithmetic unit is designed for executing mathematical analysis, such as stereoreconstruction, statistical summarization, etc. involved in processing pictorial data. The block diagram of ILLIAC III is illustrated in Figure 5.2.

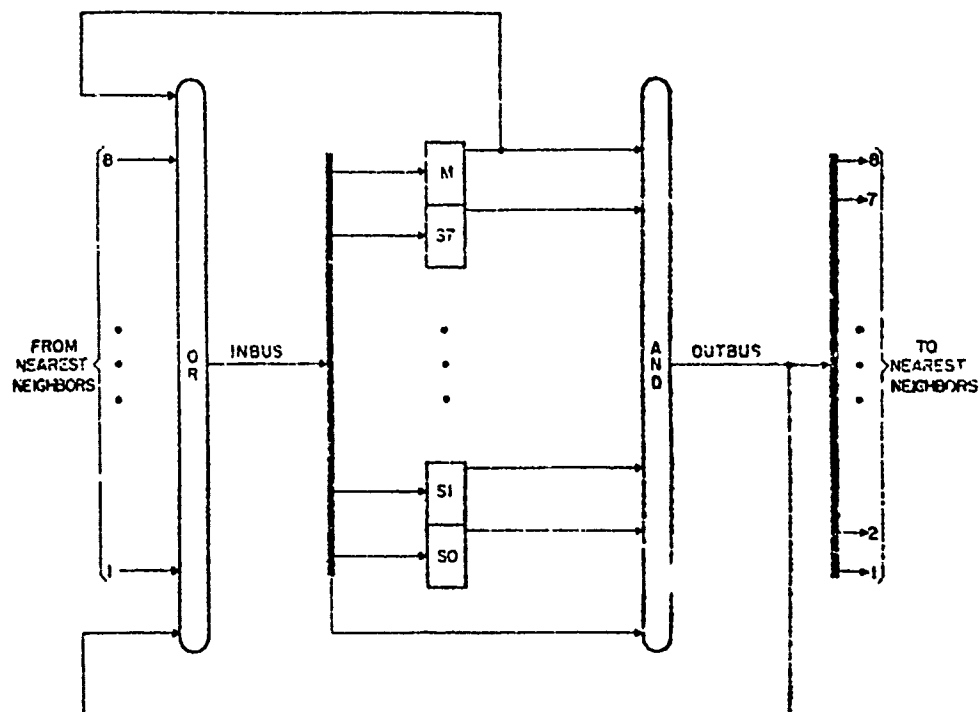


Figure 5.1. Schematic diagram of stalactite of pattern articulation unit in ILLIAC III computer.

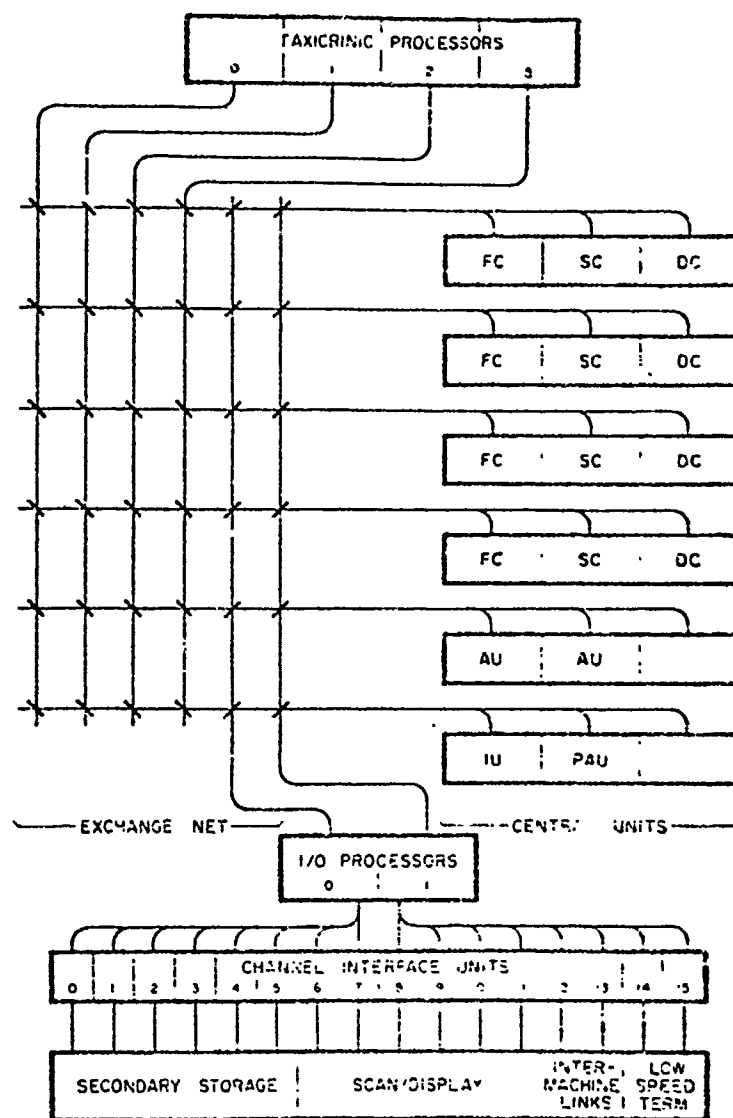


Figure 5.2. Block diagram of Illinois pattern recognition computer ILLIAC III.

One special feature of ILLIAC III for image processing is the use of an auxiliary memory. The auxiliary memory is used for the processing of images frame to frame, and for retaining intermediate partial results. The necessity of storing intermediate results of the iterative array stems directly from the manner of executing homogeneous logic transformations in the processor. The complexity of the iterative array can be greatly simplified if various digital filtering operations can be performed serially storing the intermediate results ($32 \times 32 = 1024$ bit words). For example, local track segment orientation can be designated as being predominantly horizontal, vertical, left-diagonal, or right-diagonal in four serial tests, whereas simultaneous identification requires approximately four times as much hardware [87]. Unfortunately, ILLIAC III has never been completely built.

The use of an auxiliary memory for image processing has been implemented in several computer facilities for pattern recognition research. One example is the Purdue University Advanced Automation Research Laboratory [101]. This laboratory is organized around a DEC PDP 11/45 digital computer with 32K core memory and 96 K fast secondary memory [107], two disk drives, a magnetic tape drive, two cassette tape drives, a line printer, and a CRT monitor. The system block diagram is shown in Figure 5.3. The computer is not a special purpose computer, but the auxiliary memory is a special feature for image processing. The auxiliary memory was developed for image processing this laboratory, mainly because of the limited addressing range of the 16-bit minicomputer. Therefore, the limited addressible memory is difficult to use for implementing large programs. A memory controller has been built which can access up to 2^{32} bytes of memory, which now controls 64K 16-bit words

BEST AVAILABLE COPY

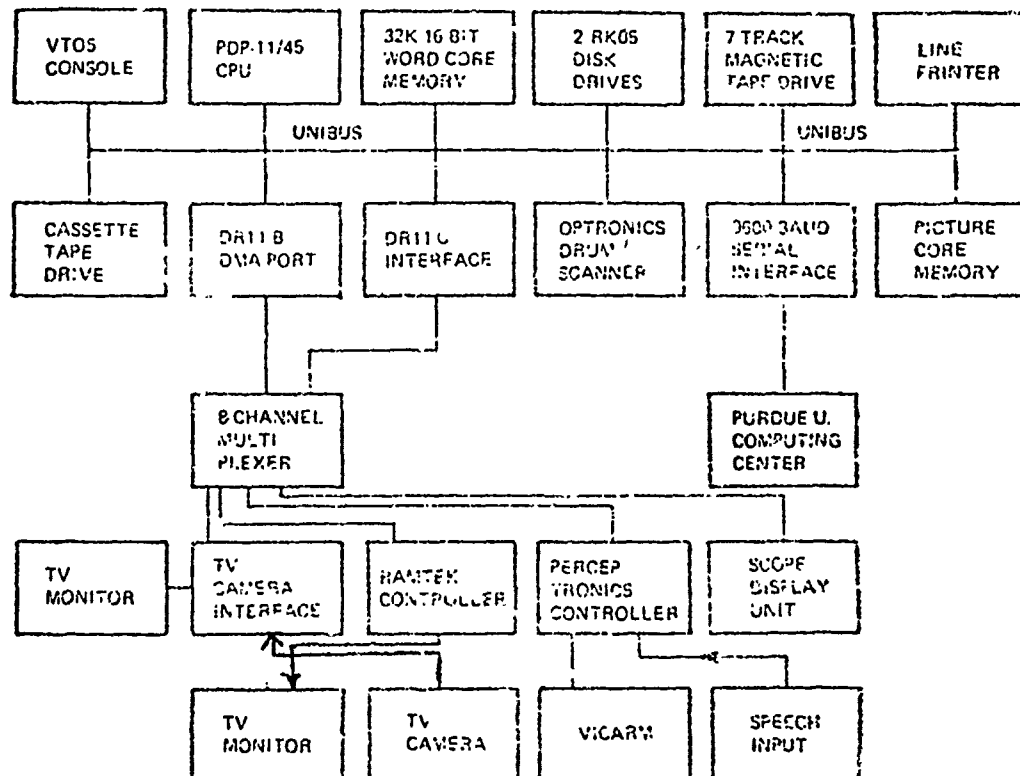


Figure 5.3. Block diagram of computer system in Advanced Automation Research Laboratory.

of core memory [107]. The controller is interfaced to the PDP 11/45 unibus through using three 16-bit registers. Two registers form an effective address of 32 bits, and the third register contains the data to be written into the memory or the data read from memory. Once the address is loaded into the two registers, data can be written into or read from memory in less than one microsecond. Since large arrays of data can be addressed without I/O to the disks, execution times improve. Also, computer programs become simpler since no sophisticated disk-core swapping software is needed. An experiment on rib identification in chest x-ray images was programmed once without using the auxiliary memory and once using this memory. On the average, the execution time improved a factor of five. Other software shows comparable execution time [101].

Digital Parallel Processors (DPP's) [89,90,91], using cellular logic, are real-time machines in which the action resulting from a program statement is simultaneous on all the points of the array. The action may be symmetrical or directional and the tessellations of various types, the most common being the square and the hexagonal ones. An example of this kind of machine is the parallel cellular logic image processor, CLIP3 [91]. The CLIP3 is comprised of an array of 192 cells arranged in a block of dimensions 16 cells (vertically) by 12 cells (horizontally). The array interconnection pattern can be either square or hexagonal. The required architecture is determined by one bit in each instruction word and is therefore under the control of the programmer. The block diagram of the cell logic of CLIP3 is shown in Figure 5.4. The Boolean processor can perform, under program control, two independent Boolean functions from its two inputs A and P. D is one output and can be regarded as the processed pattern bit corresponding to the cell. The other output, H,

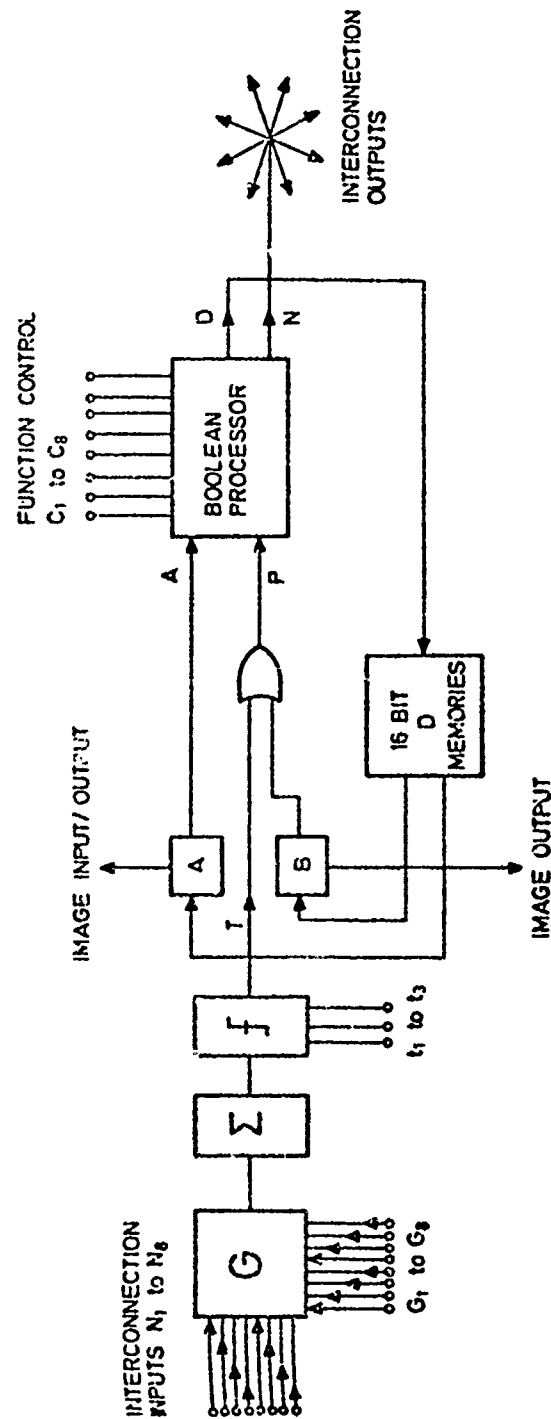


Figure 5.4. CLIP 3 cell logic.

fans out to the adjacent cells. The N inputs from neighboring cells ($G_1, G_2, \dots, \text{and } G_8$) are summed (Σ) and compared with a threshold (f). The programmer is allowed, for each PROCESS instruction, to select several inputs to be put into the summing unit and he can also choose the threshold value. The summed and thresholded output, T , is then "OR"ed with the second pattern input to form the input, P , to the Boolean processor. D outputs are addressed into any one of 16 single bits of store; buffers A and B are loaded from addressed location in the same store. The CLIP3 was actually built.

One drawback of both the ILLIAC III and CLIP3 is that the "edges" caused by moving image windows, (in ILLIAC III 32×32 windows and 16×12 window in CLIP3) are not taken care of by the computer. The computing power of the CLIP3 is obviously limited because of the fixed small number of (Boolean operator) cells. ($16 \times 12 = 192$ cells)

In order to process larger images, CLIP3 has been interfaced to a television camera through the hardwired scanning unit. This is called a hybrid CLIP3 array [92] which is shown schematically in Figure 5.5. This unit scans the 192 cell CLIP3 array across the 96 by 96 cell data-field and provides edge stores to handle the propagation signals which cross between adjacent sectors. The complete system is interfaced to a PDP 11/10 computer which serves to extend the available data and instruction storage and also provides program editing and assembling facilities.

CLIP4 [92] is the Large Scale Integrated circuit (LSI) version of CLIP3 with some small changes in the cell design. The CLIP4 uses N-MOS (N type Metal Oxide Silicon) LSI to incorporate eight processor cells in a four by two block onto one chip. The block diagram of CLIP4 cell logic is shown in Figure 5.6. The "D" store has been increased from the 8 bits

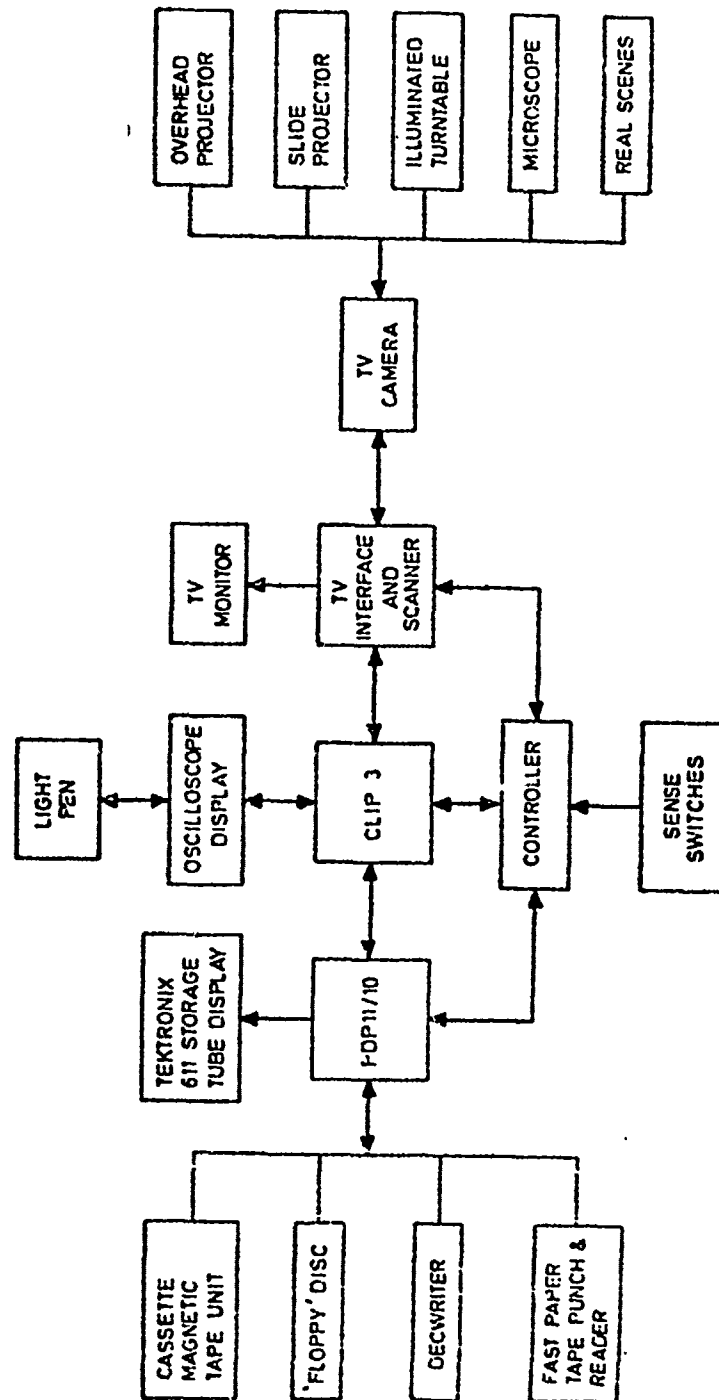


Figure 5.5. Schematic diagram of hybrid CLIP 3 system.

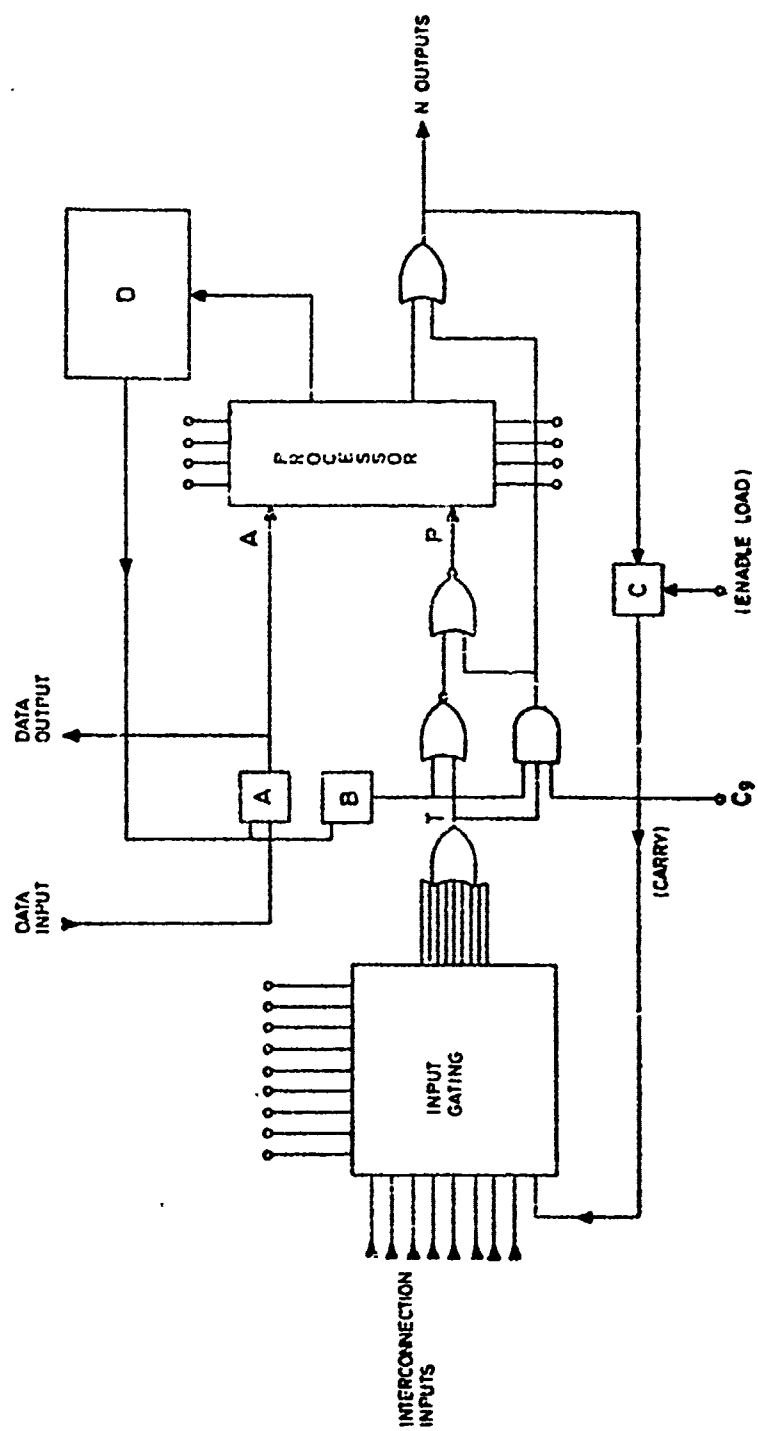


Figure 5.6. CLIP 4 cell logic diagram.

of CLIP3 to 32 bits which allow the processing of 32 grey levels. The interconnecting threshold gate has been replaced by an "OR" gate. A few extra gates and an additional buffer store have been included to provide automatic carry-in arithmetic operations.

Comparing the performance of CLIP3 and CLIP4, the processing speed of the CLIP4 cell is slower than that of CLIP3 by at least a factor of five. A single operation which required a pair of instructions (LOAD and PROCESS) in CLIP3, taking 2 μ sec, is expected to require about 10 μ s in CLIP4. Propagation from cell to cell took 0.1 μ s in CLIP3 and may take 1 μ s in CLIP4 [92]. However, the processing speed of the overall system of the CLIP4 is improved by the larger array of processors of CLIP4 (96x96 cells). The CLIP4 has not been completely built up to now.

The Parallel Picture Processing Machine (PPM) [88] is a special processor which is connected to and controlled by a conventional computer. The block diagram of PPM is shown in Figure 5.7(a). The PPM consists of the following principal parts: the processing unit, a set of nine general purpose picture registers, and a control unit. The picture registers which are shown in Figure 5.7(b) comprise nine shift registers capable of storing a picture. The main parts of the processing unit are the neighborhood matching logic (NML), shown in Figure 5.7(c), the line buffers (LB's) and neighborhood counting register (NCR) and coordinate register (COR). The demand for LB's stems from the fact that when a picture is stored in a picture register, only one picture point at a time is accessible. To be able to perform the local neighborhood operations, all nine neighborhood units must be accessed simultaneously. Therefore, two LB's are needed. Note that the speed improvement is accomplished mainly at the expense of NML. The control unit is

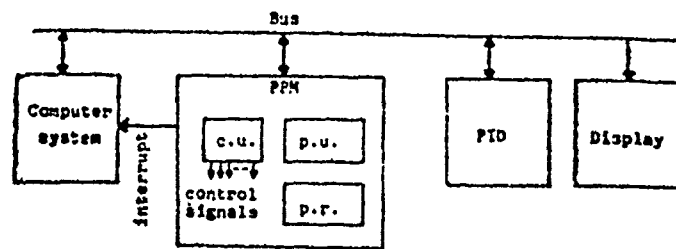


Figure 5.7(a). Block diagram of parallel picture processing machine.

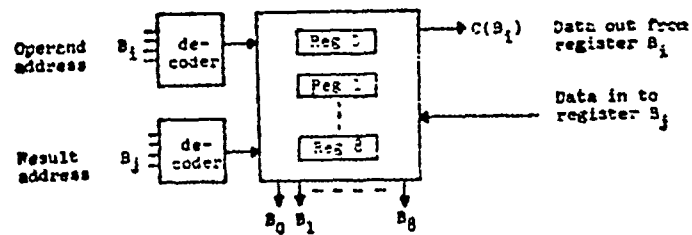


Figure 5.7(b). Picture registers.

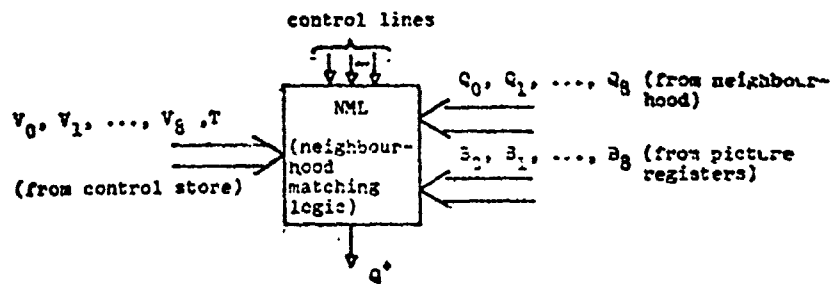


Figure 5.7(c). Neighborhood matching logic.

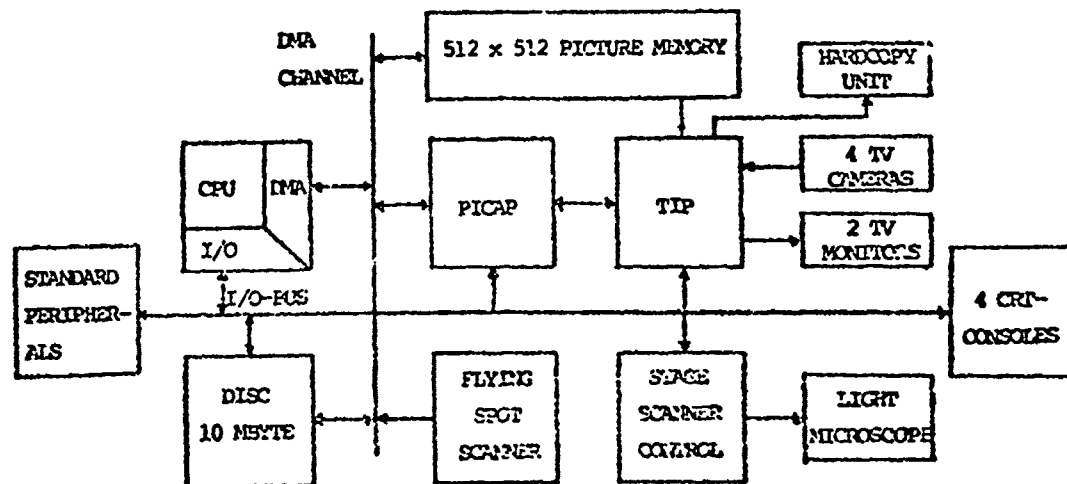


Figure 5.7(d). Block diagram of PICAP computer system.

conventional in all respects except for the writable memory where the templates of an operation to be performed are stored. The parallel picture processor in [88] has been linked with an ordinary minicomputer as part of the computation facilities of the PICAP picture processing laboratory at the University of Linköping, Sweden. The block diagram of the system is shown in Figure 5.7(d). The function of the minicomputer is twofold: (1) it controls the other devices and (2) it performs all nonpictorial data processing. PICAP which is a modified version of PPM is dedicated to the several tasks of the picture processing, such as fingerprint coding and malaria parasite detection. An important modification of PICAP has been the addition of a set of registers for the collection of measurements. The most important task of the picture processor is to produce measurements of application-dependent features. That is, to reduce the often enormous amount of information in an image to a set of feature measurements that can be handled by a conventional computer. The syntactic method for fingerprint classification can only be handled by the conventional computer in the PICAP computer of Kruse [99]. The fact that they cannot perform syntactic analysis is a drawback of the PPM's (or PICAP's).

Using the approach of distributed computation for designing a special purpose computer for image processing, the Control Data Corporation designed the Flexible Processor [97]. The Flexible Processor was developed for a large digital change detection system for concurrent processing of four channels of side-looking radar imagery. The Flexible Processor is a microprogrammable processor and uses random access memory up to 1024 words of 48 bits each for microprogram control. One special feature of the Flexible Processor is its data transmission which is

shown in Figure 5.8. Four separate balanced party-line transmission channels are available. Each 32-bit channel can initiate and receive register-buffered transfers. A scanning system allows several Flexible Processors on a party-line to communicate with each other and with peripheral devices. For output transmission, a 32-bit output channel is provided which interfaces on balanced lines. A dual internal data bus system is used to match data transfer speed to the speed of arithmetic logic. The arithmetic unit consists of an arithmetic logic unit, hardware network for conditional microinstruction execution, array hardware multiplier, specialized logic for square root and divide, and hardware priority interrupt mechanism.

Another special computer for image processing, utilizing the distributed computing approach, was developed by Toshiba Company called Toshiba Image Processing System TOSPICS [93,116]. The TOSPICS is an interactive image processing system which is a disk-based system and each operation is performed by the command input through a teletypewriter. The software system for the image processing system consists of the permanent and non-permanent resident system programs, the picture processors, the block common area, and a package of image processing programs. The system diagram of TOSPICS is shown in Figure 5.9. The special features of TOSPICS for image processing are that the image memory is commonly used in order to reduce the amount of data transmission and the parallel picture processor is constructed to perform certain programs at high speed. The parallel picture processor performs the program of spatial filtering at a speed of 1 pixel/1 μ sec. The I/O devices of TOSPICS include a unique high precision flying spot scanner by a Double Deflection

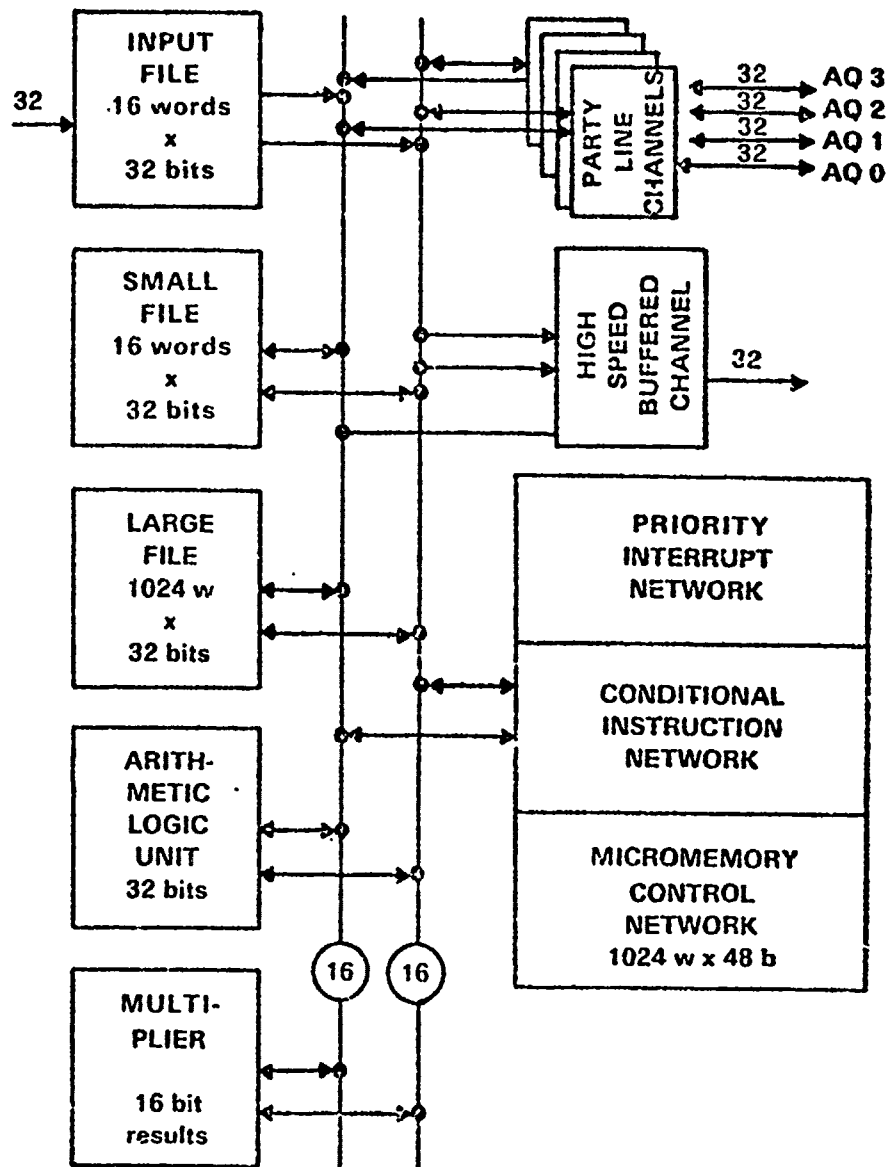


Figure 5.8. Block diagram of flexible processor by Control Data Corporation.

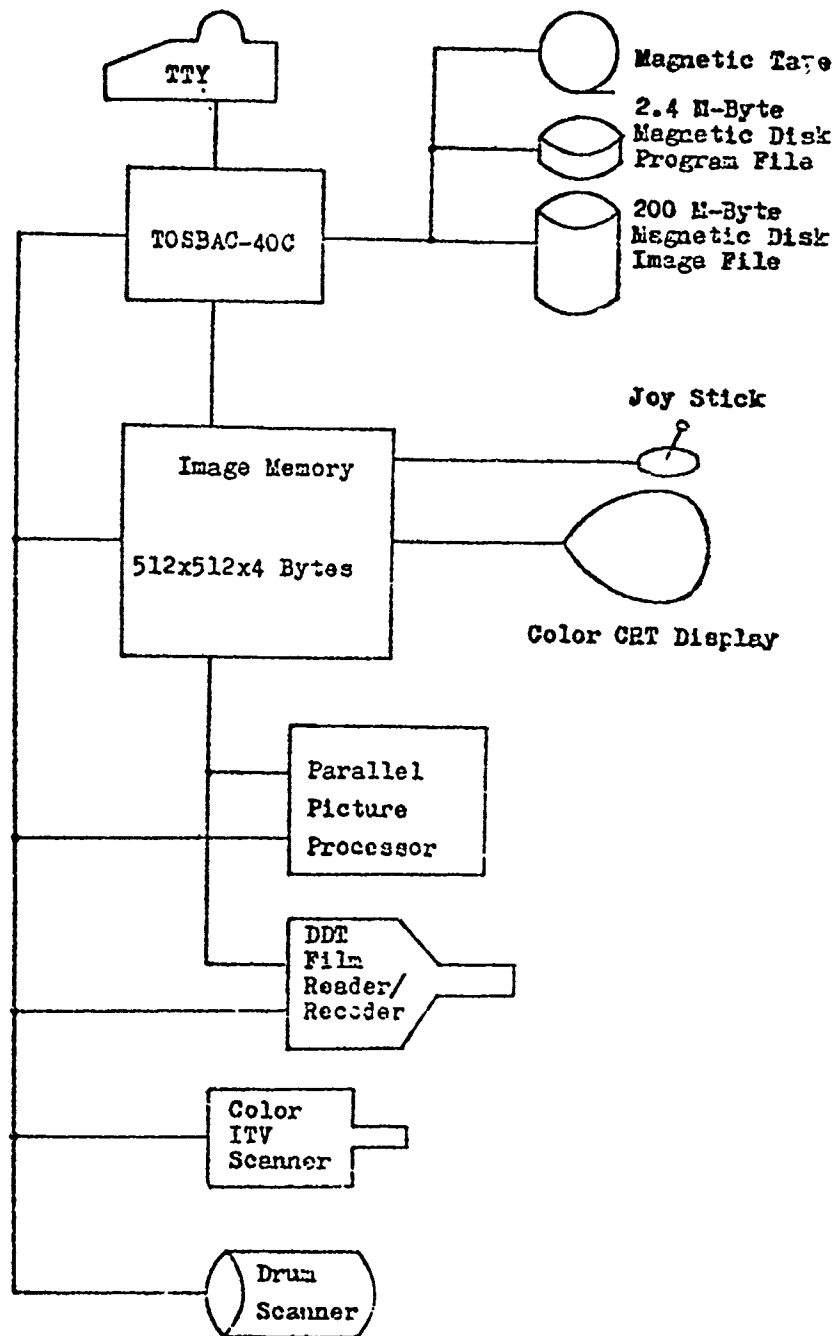


Figure 5.9. Block diagram of TOSPICS computer.

Tube (DDT) with 4096x4096 resolvable points and 32,000x32,000 addressable points.

Comparing the systems of Flexible Processors and TOSPICS, the basic ideas of parallel processing for image processing by the distributed computing approach are the same for these two systems. The difference between these two computer systems is the ways of utilizing processing elements. The TOSPICS uses the minicomputer, TOSBAC-40C, as an image processor and a parallel picture processor is attached to it. The parallel picture processor is used for certain programs, such as spatial filtering, affine transformations, histogram calculation, and density conversion [93]. The computer system of Flexible Processors uses the Flexible Processors as the processing elements. Each Flexible Processor is assigned for one task and several Flexible Processors are organized as the computer system.

The STARAN [108] computer is a parallel processing computer built by Goodyear Aerospace Company in 1972. STARAN was not designed as a special purpose computer for image processing. But this computer was utilized for the task of resampling in the field of image processing in 1977 [109]. The STARAN parallel processor is a single instruction stream multiple data stream (SIMD) processor. The previously reviewed computers, such as, ILLIAC III, CLIP4, PICAP, and TOSPICS are SIMD processor. The Flexible Processor is SISD processor. The single most important element of STARAN is the associative array, which provides content addressability and parallel instruction execution capabilities. Most STARAN computing is done within a word of associative array memory. An associative array word is normally divided into fields of varying lengths by the programmer to suit the requirements of specific programs. The values of these

fields then can be added, subtracted, multiplied, and divided within the word. The STARAN can perform the same operations as a sequential processor but with the added capability of performing these operations simultaneously on literally thousands of words in the associative processor arrays. A basic STARAN configuration contains an associative array. However, up to 32 associative arrays can be included in a single STARAN system. The block diagram of the STARAN computer is shown in Figure 5.10. The sequential controller provides off-line capabilities for assembling and debugging STARAN programs and control of STARAN error processing, diagnostic, and maintenance programs. Buffered I/O is available for tying different types of peripherals into the STARAN control memory. Also BI/O can be used to transfer blocks of data and/or programs between the STARAN control memory and host memory. The external function (EXF) logic facilitates coordination between the different elements of STARAN for special functions and simplifies housekeeping, maintenance, and test functions. By issuing external function codes to the EXF logic, elements of STARAN can control and interrogate the status of other elements. In general, the STARAN computer is powerful compared with the Flexible Processor and TOSPICS. However, only several image processing tasks (such as resampling) have been carried out on this computer. Therefore, the effective utilization of STARAN for all types of image processing is still an open question.

Consider the attributes of the bit-plane processing approach and the distributed computing approach. The bit-plane approach mostly uses Boolean operators as processors. Boolean operators work only on binary images which are not common in the real world. One way to get around this is to use several binary picture planes to represent the grey scale

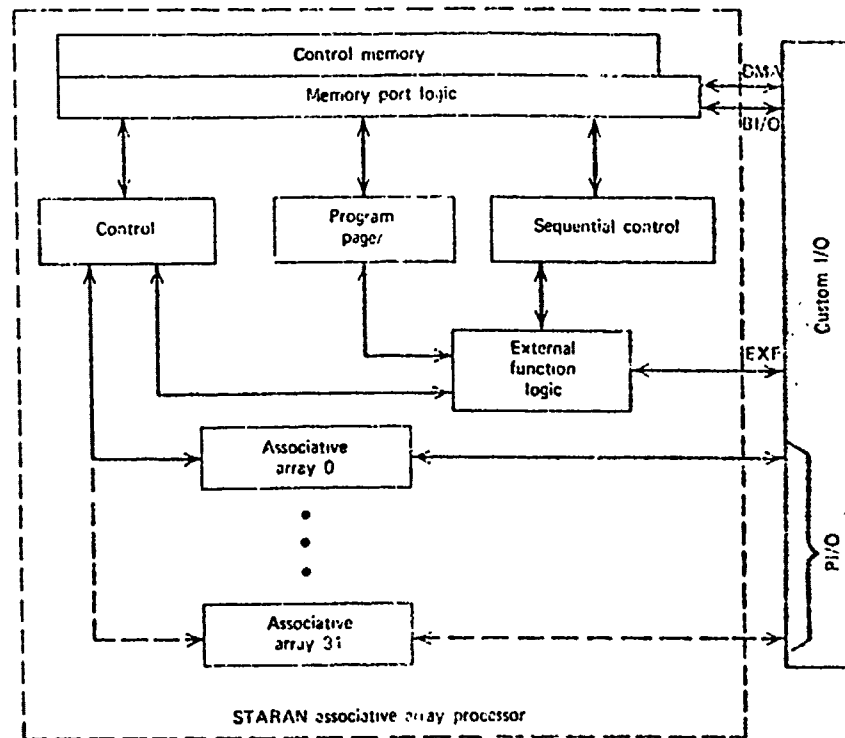


Figure 5.10. Block diagram of STARAN computer.

values of picture points. But, the complex software and additional memory requirement cause another problem and this problem limits the processing power of the processor. One of the drawbacks of the bit-plane processing approach is that the processing power of the processor may not be adequate for some of the more sophisticated image processing techniques. For example, Kruse [99] has shown that the parallel Picture Processing Machine PPM (now called PICAP) [88] to be applicable only to the preprocessing part of fingerprint classification and as stated above syntactic techniques have to be performed by the conventional computer in the PICAP system. Thus, the ability of the bit-plane approach to perform highly sophisticated, but important, techniques is, at this time, unsure. The capability of real-time processing of the bit-plane processing computers such as CLIP3, CLIP4, and PPM will be very difficult to ascertain until more complicated techniques applied to real world pictures with more grey levels such as 128 or 256, have been implemented by these computers.

After studying the feasibilities of the bit-plane processing and distributed computing approaches to the real world image processing task, we feel that the distributed computing approach is better considering the present state-of-the-art with respect to both software and hardware. In March 1977, Stone [106] indicated that the distributed computing approach is one of the future trends for general computer architecture. His remark supports our judgment on the distributed computing approach for special computer architecture for image processing. A major drawback of previous computers designed by distributed computing approach is that the system's processors are not reconfigurable. The vast varieties of sensor types, applications, and image processing

techniques, require that the image processing system (especially the parallel processor) be reconfigurable. Therefore, a generalized computer architecture which is reconfigurable under software control is proposed in the next section of this chapter for the many applications of image processing. Not only is the concept of reconfigurable ability new as special purpose computer architecture for image processing, but also the methods of exploitation of parallelism is new. In the proposed computer architecture parallelism within the task is exploited by the parallel processor. In the meantime the operations of the sequential arithmetic processor are pipelined with the parallel processor under programmer control in certain tasks which can be decomposed into pipelined processing. Therefore, parallelism and pipelining are exploited at the same time in the proposed computer architecture. The parallelism [104] used is the multiprocessing approach which subdivides each outgoing job among many identically constructed mechanisms. The pipelining, or overlap [105] processing is another multiprocessing approach which is to develop a collection of specialized mechanisms capable of working simultaneously to form a general purpose organization. The processing time of the image processing task by the proposed computer architecture will be sped up by a factor which is comparable to the amount of parallelism and pipelining existing in the image processing task of interest.

5.2 PHYSICAL ORGANIZATION AND CONTROL FLOW OF THE PROPOSED COMPUTER ARCHITECTURE

The proposed Computer Architecture for Image Processing is called CAIP and is to be designed using the most recent semiconductor technology. The physical organization and control flow are described in next subsections.

5.2.1 Physical Organization of the Proposed Computer Architecture

The physical organization of this special computer architecture for Image processing (CAIP) comprises the task management processor, the control units, the parallel processor, the sequential arithmetic processor, and the memory organization.

1) Task Management Processor (TMP) is a set of software programs which allocate the jobs to the parallel processor (PP) or Sequential Arithmetic Processor (SAP). The set of software programs include a task control program, a job control program, an input/output program, and the language translation program. The task control program provides the logical interface between the hardware and the remainder of the software system and is responsible for the allocation of jobs to the parallel processor and sequential arithmetic processor. Each task has a tag which is designated by the programmer for the identification of parallel processing or sequential processing. One part of the task control program is called the tag examination program which examines the tags on tasks and allocates the tasks to the proper processor. Following the tag examination, the initiation program, which is another part of the task control program, initiates the parallel processor or the sequential arithmetic processor. In general, the task control program performs scheduling, supervision, interruption handling, execution supervision, and clock supervision. The job control program provides a logical interface between a task and a job or between a task and the system operator. The job control program analyzes the job stream, looks at system resources, processes job execution and termination, and communicates between the system operator and the individual job program. The I/O control program provides an interface between the processing programs

and the I/O devices. The I/O control program performs I/O supervision, access routine processing, and I/O device initiation. The language translator program translates the computer language into machine codes and compiles the program to be executed.

2) The Control Units (CU's) consist of two sets of software programs. One control unit (CUPP) is for the parallel processor, and the other (CUSA) is for the sequential arithmetic processor. The CUPP and CUSA are different from conventional processors, in that they are software programs which control the operation of the parallel processor and the sequential arithmetic processor respectively. The CUPP is a control program which initiates two different sets of software operating systems, one for SIMD mode and the other for MIMD mode. The two sets of software operating systems drive the parallel processor individually upon the command of CUPP. The reconfiguration from SIMD mode to MIMD mode or MIMD mode to SIMD mode is performed by loading the operating system corresponding to the desired mode. Next, the operating system is assigned to the parallel processor (PP) by the control program of CUPP. Hence, the parallel processor operates in either SIMD or MIMD modes under the respective operating systems. Through this arrangement, the CUPP reconfigures the computer architecture from SIMD to MIMD or MIMD to SIMD. This reconfigurable capability enables this computer architecture to satisfy the large variety of applications of image processing. The operating system of MIMD mode includes scheduling routines, dynamic allocating routines, and dispatching routines. The scheduling routines schedule each job depending on job priority and facility requirements. The dynamic allocating routines take jobs set up by the scheduling routines and partition the set of processors according to the need of each

job. The dispatching routine dispatches the processors when the job terminates or some higher priority task requires processors. The operating system for the SIMD mode is on the master control unit. All the processors of the parallel processor (PP) are controlled by this master control unit and thereby an instruction is executed simultaneously on all the processors. The control unit of the sequential arithmetic processor (CUSA) controls the sequential arithmetic processor which is a microprogram-controlled bipolar processor. The control units are shown in Figure 5.11. Note that Figures 5.11, 5.12, 5.13, and 5.14 form a graphical illustration of this computer architecture (CAIP) by linking the corresponding symbols α , β , γ , and σ in the figures.

3) The Parallel Processor (PP) is an array of microprocessors. For the parallel processor, N microprocessors are connected in an array fashion. The array organization is suitable for image processing [102]. The number N is determined from the tradeoff considerations between performance and cost. The optimal number, N , varies with the task. Hence, N can only be determined at the time of implementation. In the framework shown in Figure 5.12, a set of 64 microprocessors is used to give an idea of the dimension of the problem. The control unit (CUPP) controls this set of microprocessors in SIMD or MIMD modes. This control unit enables the parallel processor (PP) to have a higher degree of flexibility and processing power. The SIMD mode utilizes a single master control unit which drives the multiple processing units (microprocessors), all of which either execute or ignore the current instruction. This SIMD mode is especially useful for the cases in which there exists (1) a large amount of independent data, (2) no restrictions preventing them from being processed in parallel, (3) a requirement for

PROPOSED ARCHITECTURE OF SPECIAL PURPOSE ARRAY PROCESSING,
IMAGE PROCESSING COMPUTER

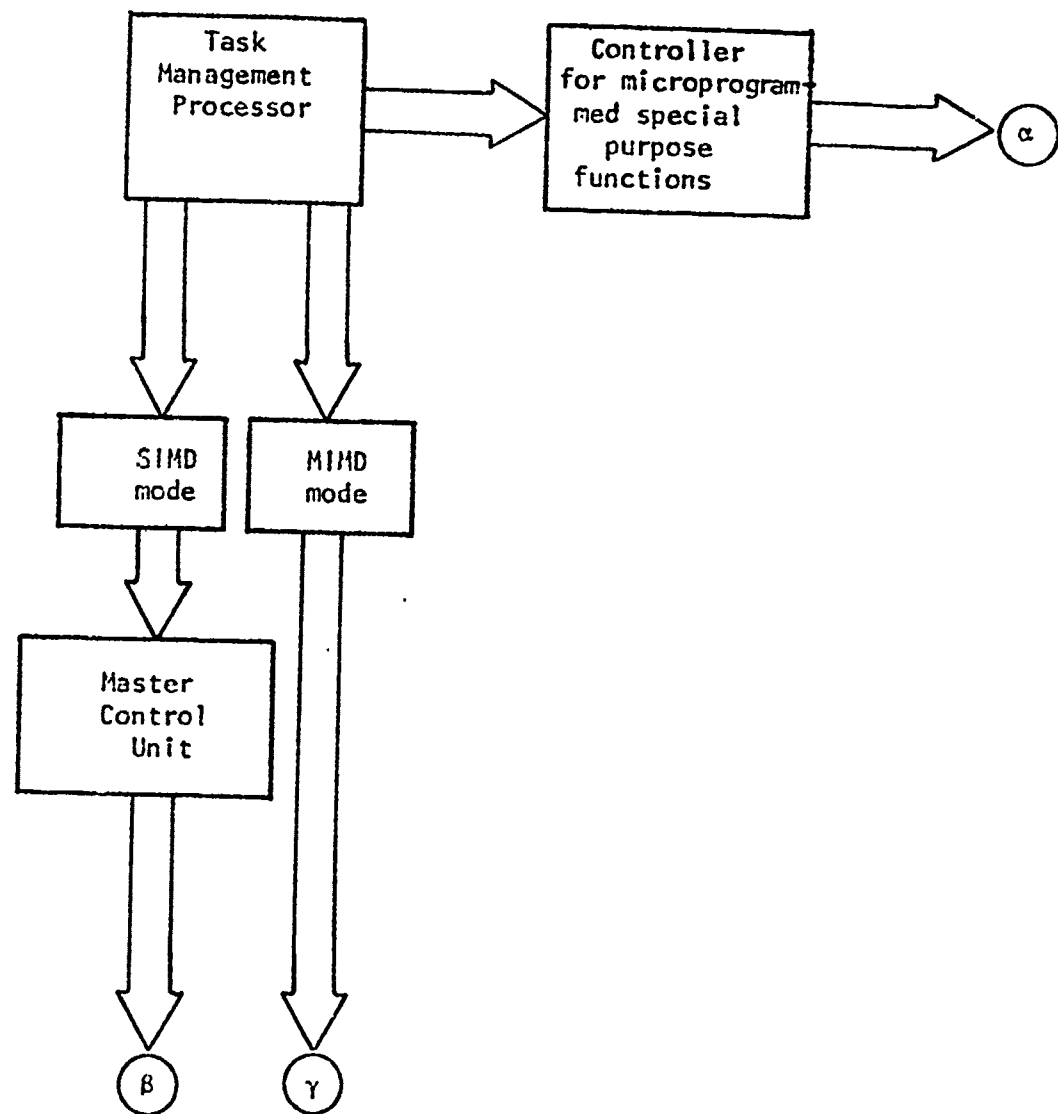
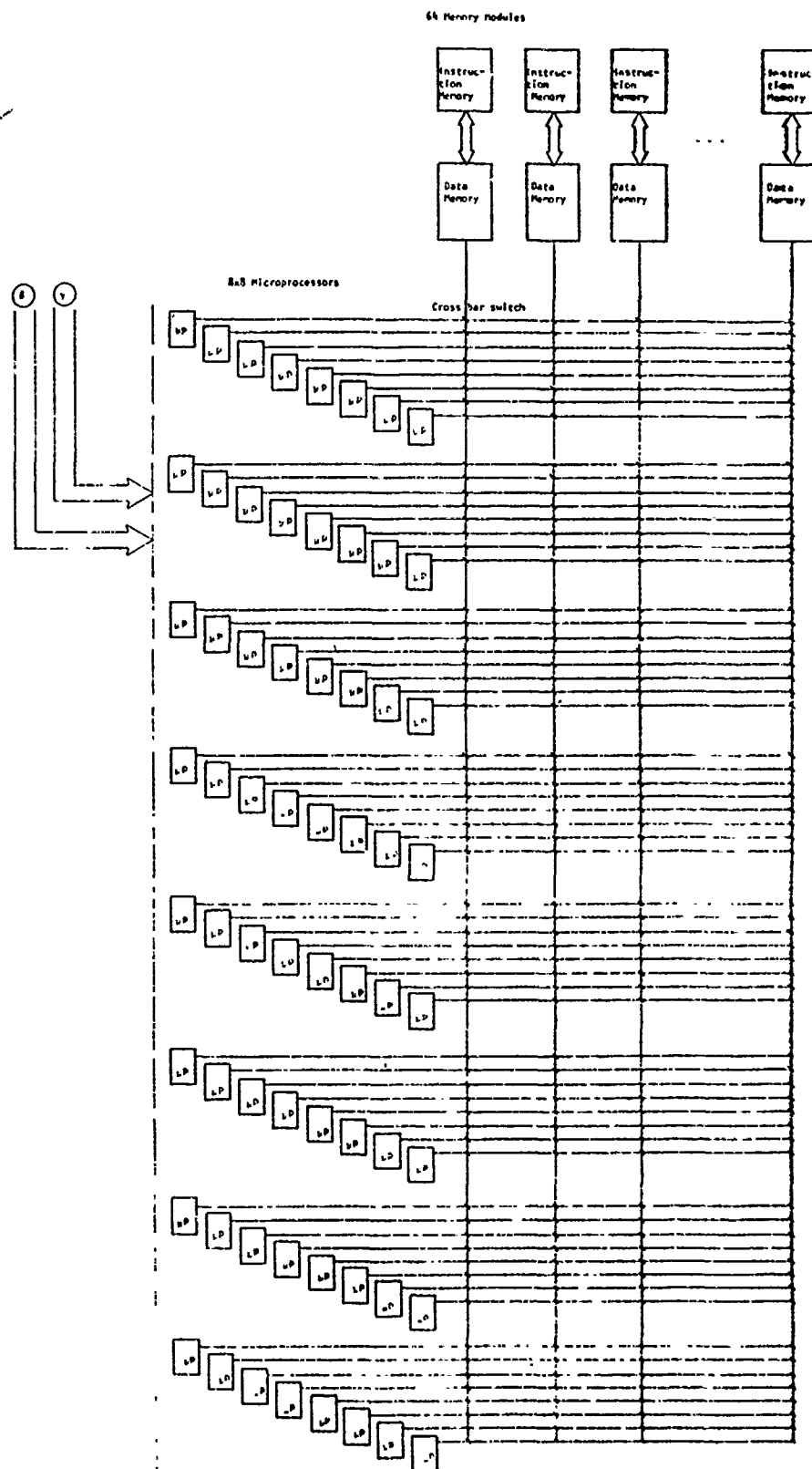


Figure 5.11. Control units of designed computer architecture for image processing.



BEST AVAILABLE COPY

Figure 5.12. Parallel processor of designed computer architecture.

high throughput, and (4) a possibility of exploiting the associative addressing selection technique. Thus, SIMD mode is suitable for the local task, which executes the same instruction on each picture element within an image window. The MIMD mode utilizes N processors and N memories where each processor follows an independent instruction stream. The parallel processor (PP) is connected by crossbar switches to an interleaved memory system which divides the ordinary memory into modules and the consecutive data are stored in different modules. The interleaved memory system is used because the bandwidth is greater than a conventional memory system. The interleaved memory system is more appropriate for parallel processing than a conventional system, in which data can only be accessed one at a time [94]. The cost of crossbar switches is high, since every processor can communicate with the other processors and memory modules. Depending on the image processing tasks specified by the user, the interconnection among processors and memory modules needed for these tasks can be studied, the alternatives for the crossbar switches might be selected.

4) The Sequential Arithmetic Processor (SAP) is a microprogram-controlled processor. Mini and micro-computers are not used here because user microprogrammable capability and bipolar processor are not furnished by usual mini or micro-computers. The Sequential Arithmetic Processor (SAP) is a bipolar processor which is a processor built by bipolar semiconductor technology and usually has the bit slicing capability. The bipolar microprogrammable processor permits the designer to define his own instruction set and the associated hardware architecture to achieve special capabilities, such as, variable word

length capability or to perform an application with the highest efficiency. The bipolar processor expands the CPU word length by cascading the needed number of bit-slice microprocessor components. This variable word length capability of SAP makes it more general and powerful than other microprocessors. Along with this processor, a microprogram memory is needed to store the microprograms of certain programs frequently in use. For the microprograms of image processing techniques, no instruction fetching is required because of the coding of the microprogram. The microprogram capability saves processing time according to the ratio of instruction fetch time to total execution time. The Sequential Arithmetic Processor is shown in Figure 5.13.

5) The Memory Hierarchy Organization is the memory system for SAP. The picture is stored on a magnetic disk memory which is more economical than core memory, but memory access time is long. The memory access time of the bipolar memory is faster by a factor of 100 than disk memory [111]. Therefore, a semiconductor bipolar memory is connected to the disk memory as working memory space and buffer. The hierarchy organization is as follows: the picture area which is to be processed is loaded onto the bipolar memory from the disk memory, then the processor gets the data (picture points) from the bipolar memory thus allowing extremely fast memory access time. In order to avoid being delayed by the loading time from disk to bipolar memory, a Bipolar Memory Buffer (BMB), is used. While the processor is reading the data into the bipolar working memory space, the next picture area is loaded on the Bipolar Memory Buffer (BMB). Thus, this memory preloading makes the data always ready in the fast-access bipolar memory. This memory hierarchy organization

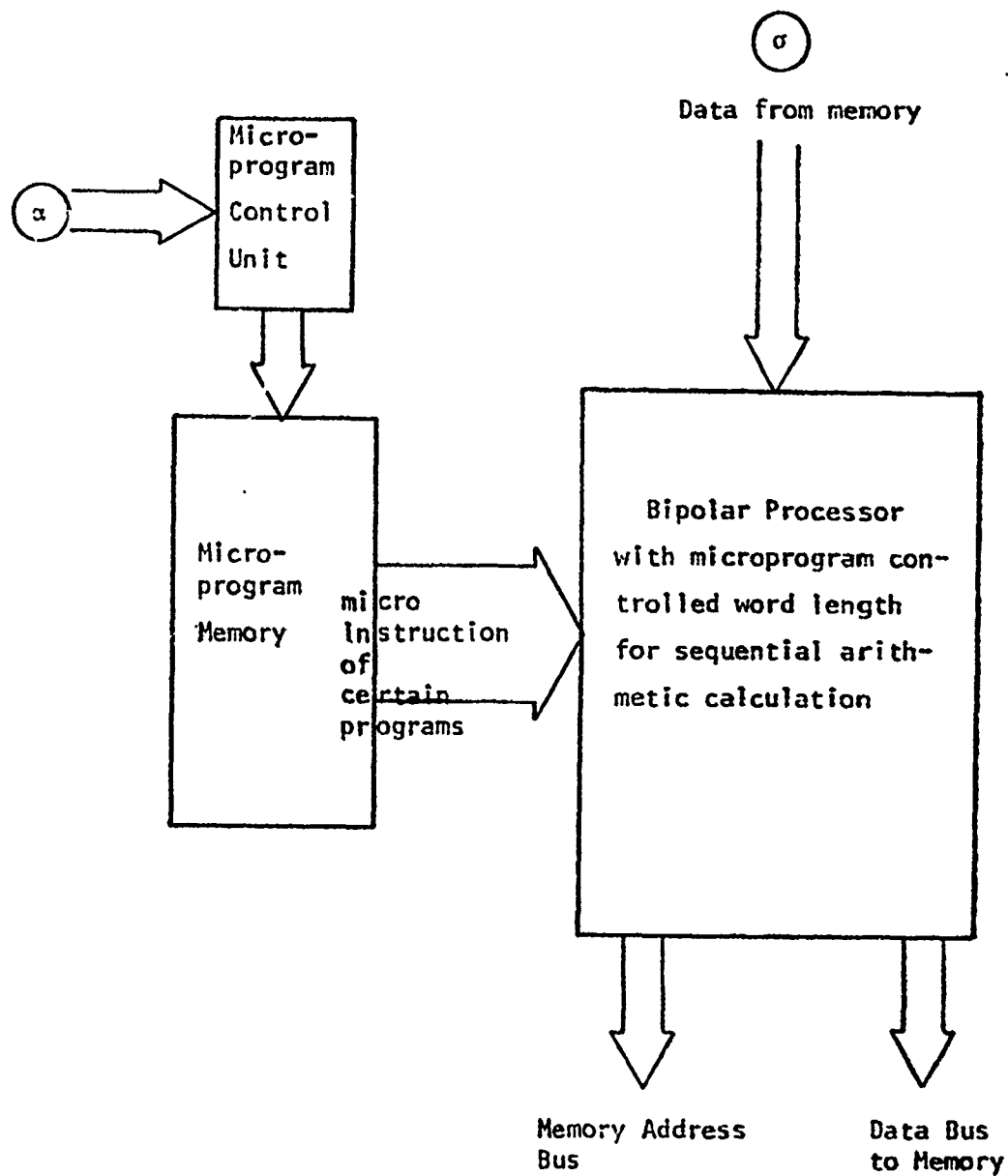


Figure 5.13. Sequential arithmetic processor of designed computer architecture.

is illustrated in Figure 5.14. This memory hierarchy organization is different from the memory of TOSPICS [116] and the image core memory of the 11/45 [107]. The approach of the design of the memory system of TOSPICS [116] was to employ a large memory storage to reduce the memory loading time. The approach in designing the secondary core memory of the PDP 11/45 [107] used primarily for image storage, was to utilize this secondary storage as a buffer memory as in the auxiliary memory concept of the ILLIAC III [100]. However, the secondary core memory of the PDP 11/45 is not randomly accessible. Each time an image datum is needed the user's program has to request that datum to be loaded to the image core memory. The proposed memory hierarchy loads a large block of data onto the Bipolar Buffer Memory as image processing necessitates operations on large blocks of data, this ability of the CAIP through its BMB provides a marked advantage for image processing. Once the large block of data has been loaded onto the bipolar memory, any individual data point can be randomly accessed and individual loading from primary to secondary storage is not needed. Thus, the user's software becomes simpler by virtue of this proposed memory hierarchy organization. The bipolar memory is used here because the bipolar memory is the memory device with the fast fetch-time in present technology [111]. The disadvantages of using bipolar memory are the higher cost and power needed than the core memory.

5.2.2 Control Flow of the Proposed Computer Architecture

The control flow for the proposed computer architecture, CAIP, is shown in Figure 5.15. The Task Management Processor (THP) allocates jobs to the parallel processor (PP) or the sequential arithmetic processor

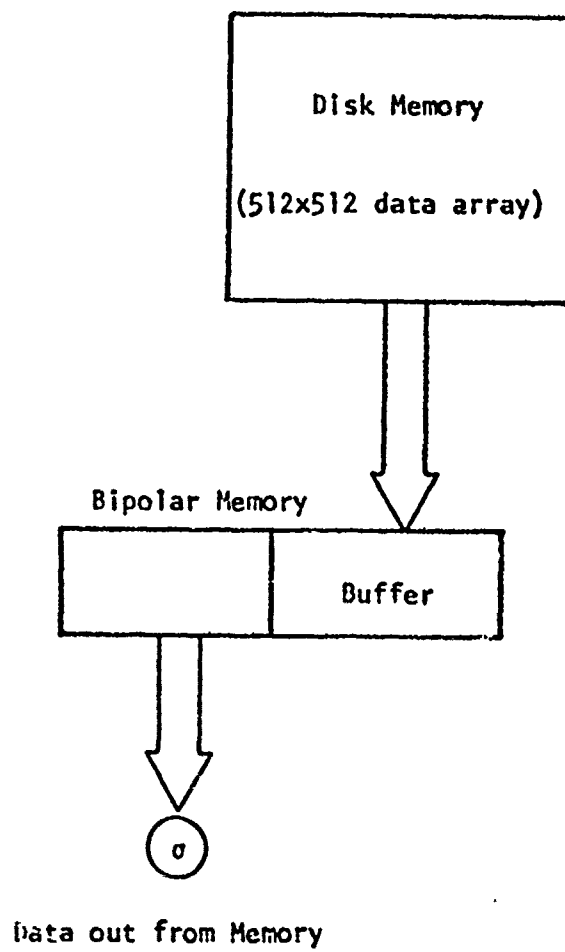


Figure 5.14. Memory hierarchy of designed computer architecture.

CONTROL FLOW OF DESIGNED COMPUTER ARCHITECTURE

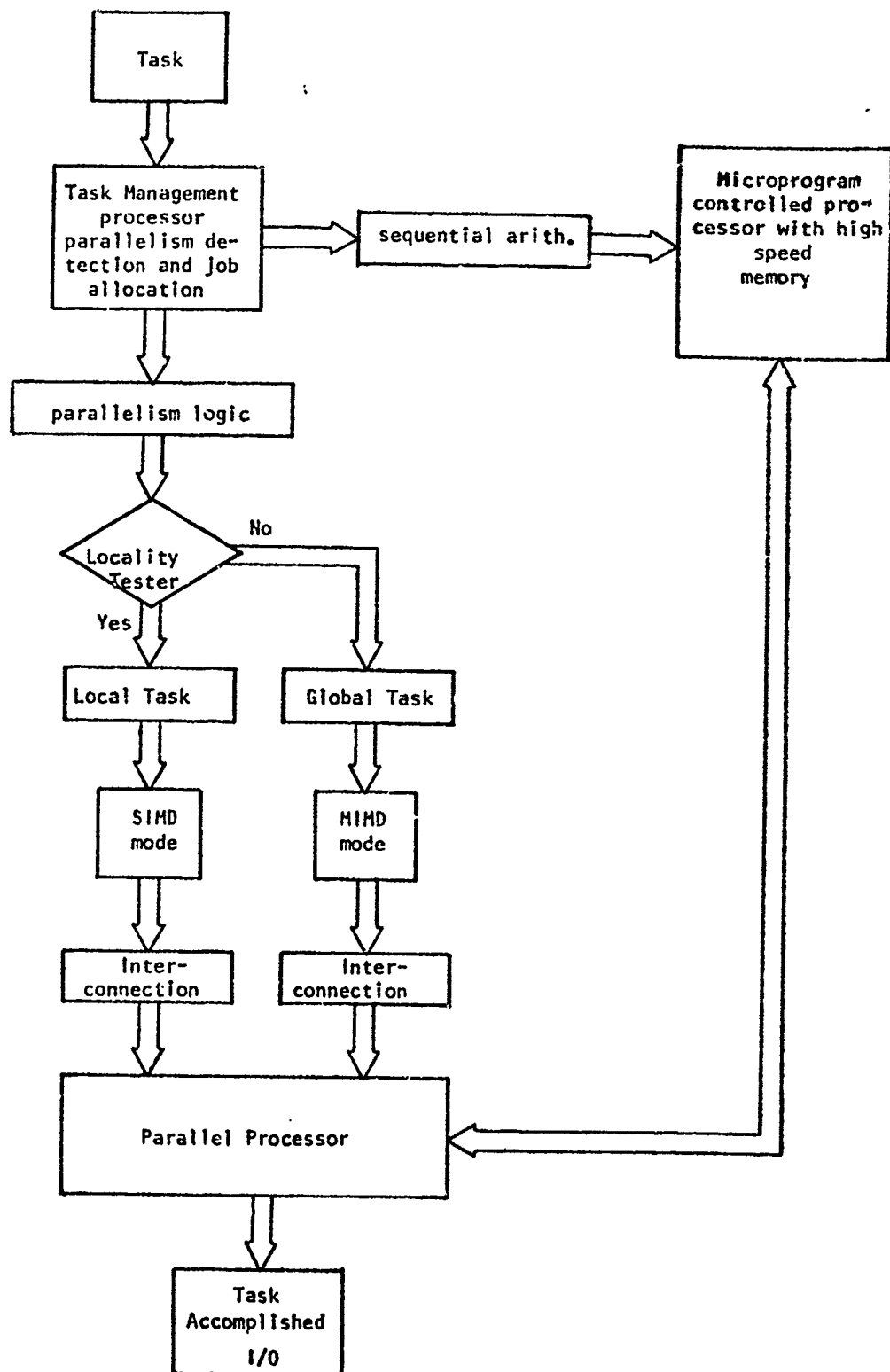


Figure 5.15. Control flow of designed computer architecture for image processing, array processing.

(SAP) by means of the tag examination routine. The user's program provides a flag to the parallel processor which indicates whether the task is local or global. The locality tester examines the flag and initiates the SIMD or MIMD mode. The SIMD mode is appropriate for local tasks. In this mode a single instruction is executed simultaneously on the image points. The local task means that the instruction is executed on individual data within an image window, such as calculating the histogram of an image window. For global task, the MIMD mode is employed. The global task means that part of the task is performing one kind of operation and other part of the task is performing another kind of operation. For example, in the task of evaluating textured and nontextured areas, some processors perform second order statistical texture analysis on certain window and some processors perform first order mean vector analysis on the corresponding windows. The global task comprises of two different natures of subtasks. The outputs from the Sequential Arithmetic Processor (SAP) communicate with the parallel processor. Therefore, the SAP may support the PP and vice versa. Parallelism of task is exploited by the parallel processor (PP) to obtain high speed performance. In the meantime, the operations of the Sequential Arithmetic Processor (SAP) are pipelined to the parallel processor under program control in certain tasks which can be decomposed into pipelined processing. Therefore, the control flow of this architecture exhibits both parallelism and pipelining. This arrangement has not been incorporated into any of the existing systems discussed in section 5.1. Since two types of multiprocessing, parallel processing and pipeline processing, are exploited simultaneously, this contributes to high speed performance in the proposed computer architecture.

5.3 ANALYSIS

5.3.1 Performance and Cost-Effectiveness Tradeoffs

The Parallel Processor (PP) of the proposed computer architecture consists of an array of microprocessors which are the processing elements. In determining the optimal number, N , of microprocessors for the parallel Processor (PP), a systematic procedure is needed. Such a procedure follows: In designing the parallel processor, as the number of processing elements (microprocessors) increases, the number of data points processed by each processor decreases and processing speed increases, but the scheduling overhead also increases. Therefore, the processing speed improvement reaches a saturation point at a certain number of processing elements. So it seems that the number of processing elements corresponding to the saturation point would be a good choice. However, the answer is not that simple, as cost-effectiveness is an important factor in the feasibility of a computer architecture. Thus, the costs, such as hardware and software costs of the parallel processor (PP), need to be considered. Hardware cost usually involves the hardware purchased and the cost of physical construction of the system. Software cost refers to development of the operating systems and software supports. The best choice of optimal number of processing elements is obtained by evaluating the performance improvement and cost increment on different image processing tasks. The optimal number directly depends on the specifications which are given by the user. Let us use the following hypothetical example for illustration, if the processors' cost increases with the increase in the number of processing elements (microprocessors), and the software cost increases more rapidly than the processors' cost as the number of processing elements increases, the performance curve becomes saturated at

80 processors for task A, and 70 for task B shown in Figure 5.16. N should be in between 70 and 80. The processors' cost increases more or less linearly with the number of processors (if fewer than 100 processors are bought). But, as stated above, the software cost increases more rapidly than the processors' cost. If the software cost increases sharply at 60 processors, as in Figure 5.17, the optimal number of processors, considering cost and performance trade-off, is between 60 and 80. Depending on the specifications of the user, if the concern is more for performance than cost, then a number near 80 is chosen. If the user is more concerned about cost, then a number near 60 should be chosen.

5.3.2 Implementation of Statistical Methods

As discussed in Chapter 2 during recent years, a number of image processing algorithms have been developed [2,83]. In this section, some image processing techniques are discussed in terms of the proposed computer architecture, CAIP, to exemplify the operations of the special computer for image processing.

Statistical texture analysis has been an important topic in the field of image processing [9,72,73]. The texture analysis technique in Chapter 4 consists of histogram equalization, and texture feature measurement. In applying the proposed computer architecture, CAIP, to such a texture analysis technique, the task of texture analysis is assigned a tag P (which stands for parallel processor task) by the user's program. (Tag S stands for sequential arithmetic processor task). The subtasks of texture analysis, such as histogram equalization and texture feature measurement, are designated by the flags SI (which denotes the SIMD mode for the task), and MI (which denotes the MIMD mode for the

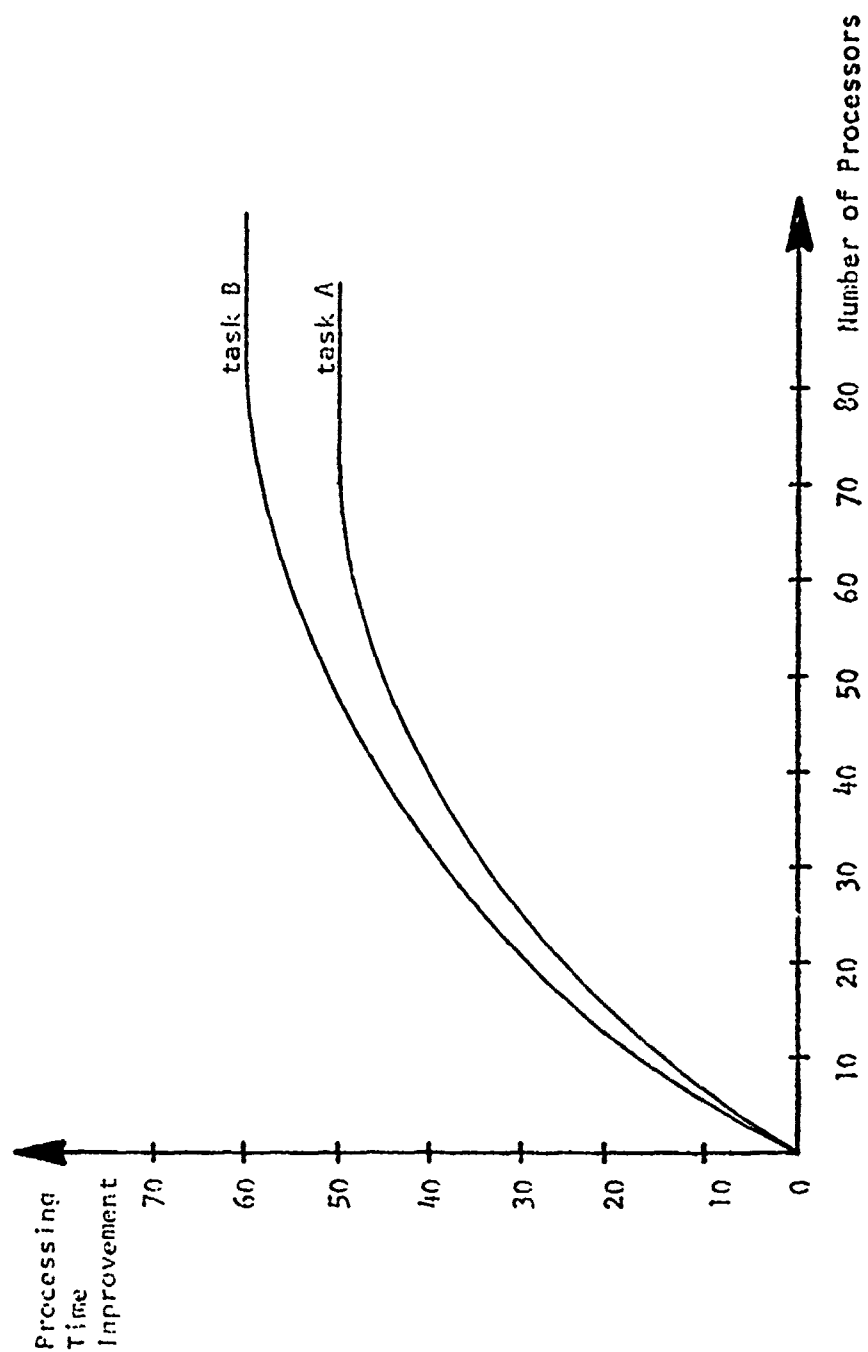


Figure 5.16. Performance Improvement versus Increments of number of processor.

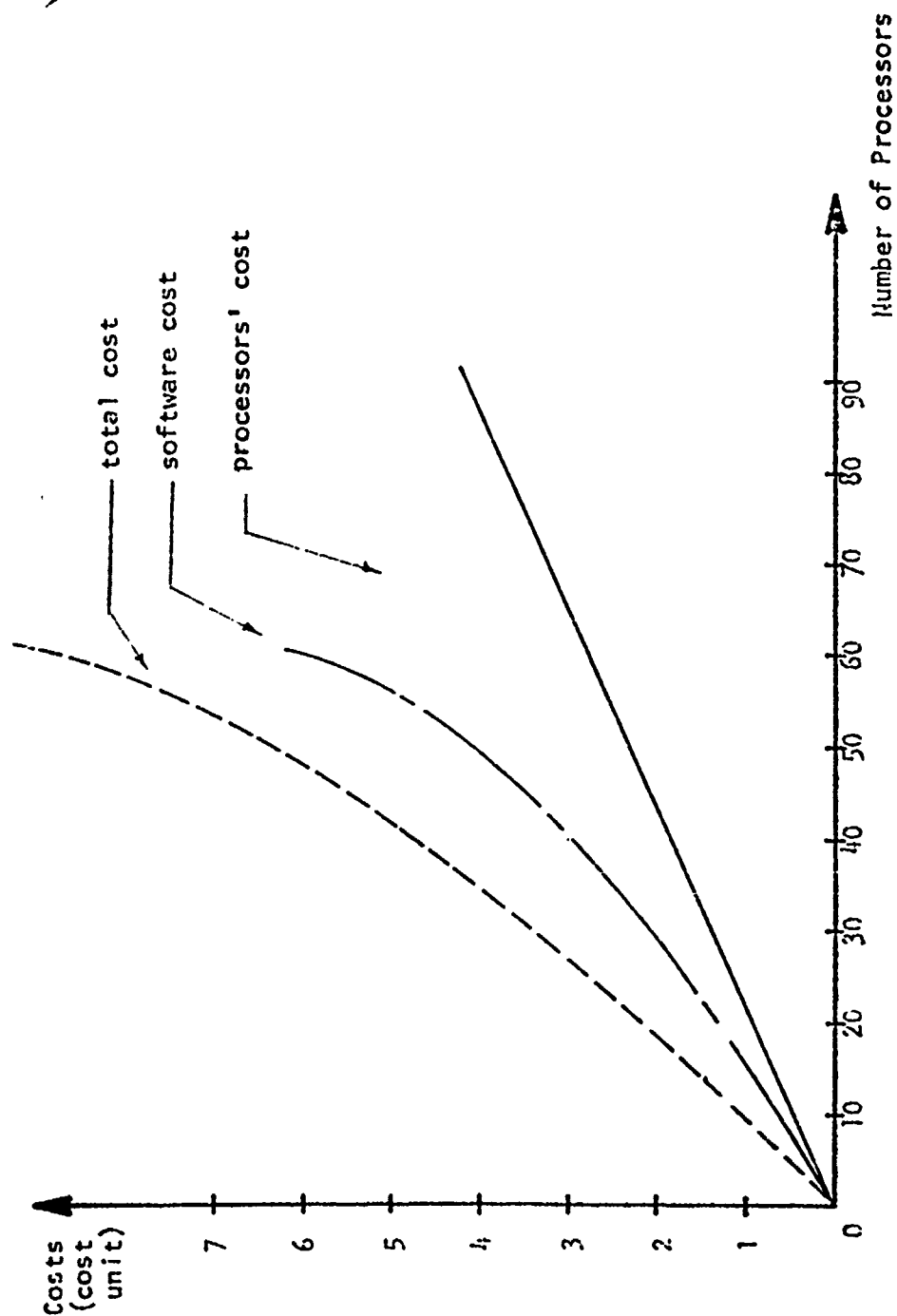


Figure 5.17. Cost estimation versus number of processors.

task). Assuming the size of the image is $M \times M$. The texture analysis task is allocated to the parallel processor (PP) of the CAIP by the tag examination routine of the Task Management Processor (TMP). The Control Unit of the Parallel Processor (CUPP) finds the flag, SI, for histogram equalization and loads the operating system of the SIMD mode. Each processor of the N microprocessors then calculates the histogram of an

$\frac{M}{\sqrt{N}} \times \frac{M}{\sqrt{N}}$ picture window. The outputs of the N processors are then put together to obtain the equalization result of the final histogram. The CUPP keeps the parallel processor (PP) in the SIMD mode after examining the flag SI of the texture feature measurement task. For example, if the 88×88 picture discussed in Chapter 4 is to be processed, each processor will process the co-occurrence matrix of the window of 11×11 pixels. The variability texture feature measurement is calculated by each processor as the texture value for the center cell (4×4) of that window. The mapping of the array of processors to the image points is shifted four pixels and repeats the texture feature measurement task. When this shift reaches the right edge of image, the mapping is shifted four pixels downward and the process is repeated from the left-most column of the image. This process continues until the texture values of all the picture points are obtained.

5.3.3 Syntactic Methods and Parallel Processing

As has been pointed out previously, syntactic methods for image processing have increased in importance for certain applications. Previous special computers such as the PPII and the PICAP are unable to process syntactic methods by parallel processing [99]. However, the

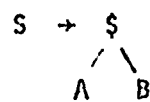
proposed CAIP furnishes the capability of parallel processing of syntactic algorithm. In using the parallel processor for syntactic methods, parallel parsing schemes are most desirable as they can utilize the capability of the parallel processor. Unfortunately, the research in parallel parsing schemes is very limited. In this section, we introduce the parallel parsing of tree languages and explore the parallel parsing of the parallel context-free languages [103].

1. Tree Languages and Parallel Processing

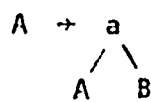
The Parallel Processor (PP) of the proposed computer architecture for image processing can be applied to tree grammar parsing. The parallel parsing procedure of a tree grammar is described below. The task of tree grammar parsing is designated by a flag P and tag S1 by the user's program for the effective utilization of the facilities. Through the control unit of the computer architecture, the parallel processor is put into the SIMD mode for the task of tree grammar parsing. If a production rule of the tree grammar which is applied to parse the language has k branches, then each of these k branches has a nonterminal. Each processor of the Parallel Processor (PP) is assigned by the user to parse one nonterminal. This procedure is applied to consecutive parsing of the language until a final parsing result is achieved. If the parsing is successful, then the language (pattern) is accepted. If the parse fails, then the language (pattern) is rejected.

For example, the tree grammar is $G_t = (V, r, P, S)$, where $V = \{S, a, b, \$, A, B\}$, $V_T = \{+a, +b, \$\}$, $r(a) = \{2, 1, 0\}$, $r(b) = \{2, 1, 0\}$, $r(\$) = 2$ and $P: [2]$

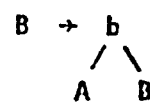
rule 1:



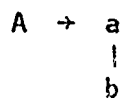
rule 2:



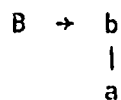
rule 3:



rule 4:



rule 5:

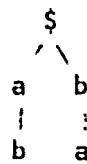
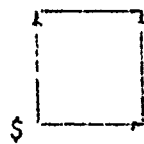


rule 6:

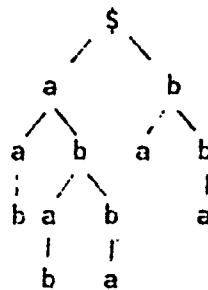
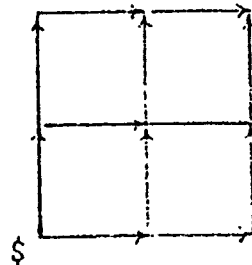


This grammar generates such patterns as

(1)



(2)



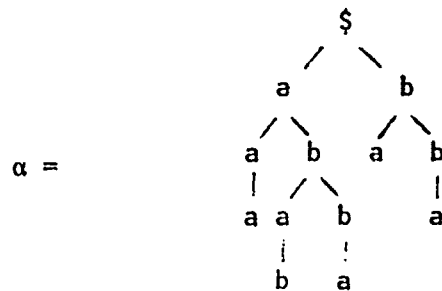
In order to perform parallel parsing of the tree language, processors need to be assigned. First, the depth "d" of the tree is defined as the number of the levels of the tree. The depth of the tree in (1) is 2 ($d=2$) and the depth of the tree in (2) is 4 ($d=4$). The maximum number of branches for all the tree grammar rules is easily obtained by checking the values of r in the grammar $G_t = (V, r, P, S)$ and this number is called n . The relationship between n and r is that n is the maximum of the values of the r 's. The number of needed processors in the parallel processor (PP) is $(d)^m$. This procedure is performed for the worst case

protection concept which allows the maximum number of branching in each level of the tree parsing. If the number $(d)^m$ is greater than the number of processors of the parallel processor (PP), there are two solutions: one is the static priority procedure, and the other is the dynamic priority procedure. The static priority procedure has a fixed priority rule for assigning the available processors to the proper subtasks. The fixed priority rule for the parallel tree parsing scheme is to assign the k left-most branches the equal priority in each parsing stage. The number, k , is the number of available processors of the parallel processor (PP) for each parsing stage. At parsing stage one, the k left-most branches are parsed first, based on the highest priority rule. At parsing stage two, the k left-most branches (nonterminals) of stage two then have the highest priority to be parsed. Thus, the k available processors are assigned to parse these nonterminals.

In the dynamic priority procedure, a dynamic priority rule has to be established at each stage of parsing in order to determine which subtasks have the highest priority. For example, the k left-most priority could be assigned first, then the k right-most priority assigned next at the request of the user. However, in parsing the tree languages, all the individual branches (nonterminals) have to be parsed to get the nodes (terminals), therefore, the static priority procedure is better. The dynamic priority procedure would only be used in special cases, such as the case in which only a partial parsing result is of interest.

Using the example given above to illustrate the proposed parallel parsing scheme for tree languages and to compare the parsing result with conventional parsing scheme for tree languages, if the depth of the tree

to be parsed is, at most, four - the maximum number of processors needed is easily calculated to be $(4)^2 = 16$. In the parallel parsing of the input tree α ,



The task is allocated to the parallel processor (PP) by the P flag found by the Task Management Processor (TMP) and the parallel processor (PP) is placed in the SIMD mode by CUPP. The procedure is graphically illustrated in Figure 5.18. At the parsing of depth one, one processor parses the language by the rule 1. At the parsing of depth two, two processors are assigned. The number of processors needed for the parsing of depth k is automatically determined by the number of branches obtained from the parsing of depth $k-1$. The number of branches obtained from depth one in our example is two. Thus, two processors are needed, one for the parsing of the branch starting from nonterminal A and the other for the parsing of the branch starting from nonterminal B. Grammar rules 2 and 3 are simultaneously applied by the two processors to parse the two branches of the tree α starting from A and B respectively. At the parsing of depth three, four processors are needed. Two processors simultaneously parse the branches of A and B which are the result of parsing rule 2 in the depth two. The parsing rules for these two processors are rules 4 and 3, respectively, to nonterminal A and B. The

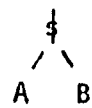
other two processors simultaneously apply rules 6 and 5 to parse the branches of A and B, which are the result of parsing by rule 3 in the depth two. At the parsing of depth four, since there are only two non-terminals A and B left from the parsing of depth three, two processors are assigned to simultaneously parse the branches A and B by rules 4 and 5 respectively. Thus, the parallel parsing is completed and the tree is accepted. The parsing of the same tree language by the conventional sequential parsing scheme is shown in Figure 5.19. At each parsing stage, only one nonterminal can be parsed by the parser. At parsing stage one, grammar rule 1 is applied. Rule 2 is applied at parsing stage two. Then the rules 4,3,4,5,3,6, and 5 are applied to parsing stages 3,4,5,6,7,8, and 9 respectively. Nine parsing stages are needed for the parsing of the same tree as shown in Figure 5.19. It can be seen in Figure 5.18 that only four parsing stages were needed for parallel tree parsing scheme. Thus, in this example there is a saving of over 50% in parsing stages, and, therefore, a corresponding saving in time by utilizing this parallel parsing scheme on the proposed computer.

2. Parallel Context-Free Language and Parallel Processing

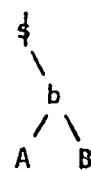
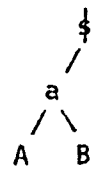
The parallel context-free language was defined by Siromoney and Krithivasan in [103]. The definition of a parallel context-free language is a language generated by a context-free grammar in which the manner of applying the grammar rules is restricted as follows: if a nonterminal occurs more than once in a sentential form, then every occurrence of the nonterminal is replaced at the same time by the same rule.

The parallel processor (PP) of the proposed computer architecture for image processing (CAIP) performs the parsing of a parallel context-free language in the SIMD mode. From the definition of a parallel

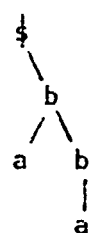
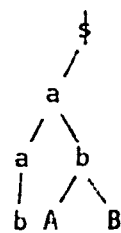
Parsing Stage 1



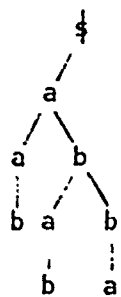
Parsing Stage 2



Parsing Stage 3



Parsing Stage 4



(Final Result)

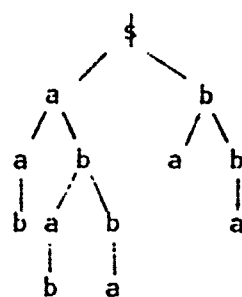
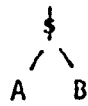
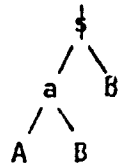


Figure 5.18. Parallel tree parsing procedure.

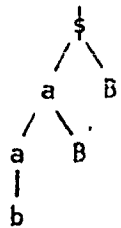
Parsing Stage 1



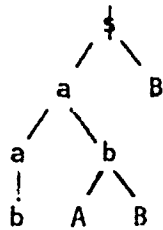
Parsing Stage 2



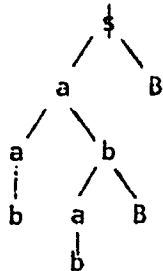
Parsing Stage 3



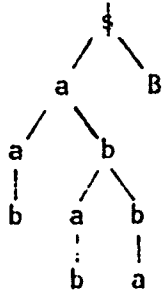
Parsing Stage 4



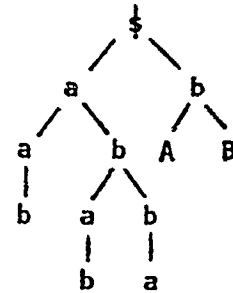
Parsing Stage 5



Parsing Stage 6



Parsing Stage 7



Parsing Stage 8

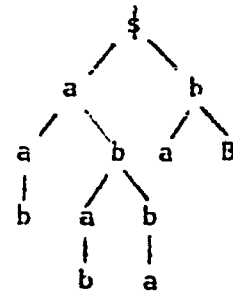
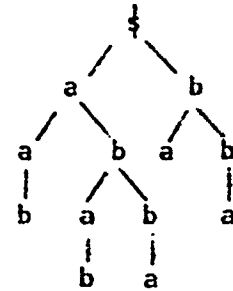
Parsing Stage 9
(Final Result)

Figure 5.19. Conventional tree parsing procedure.

context-free language, the number of needed processors of the parallel processor is equal to the maximum number of occurrences of a nonterminal in all sentential forms. A processor is assigned under programmer control to each nonterminal when it occurs simultaneously with the same nonterminal in the derivation. These processors perform the parsing of same grammar rules on these nonterminals.

For example, the task of parsing a parallel context-free language is assigned the flag P and tag S1 which initiate the SIMD mode for the task. The parallel context-free language is $L(G) = \{a^{2^n} | n \geq 0\}$ [103]. The parallel context-free grammar is $G = (V, I, P, S)$ where $V = \{S, a\}$, $I = \{a\}$, and $P = \{S \rightarrow SS, S \rightarrow a\}$. If the languages to be parsed are aa and aaaa. The maximum number of occurrences of nonterminals in all sentential forms is four which comes from aaaa (nonterminal ssss). Thus, the number of needed processors is four. At the first stage of parsing of the language aaaa, one processor is assigned to parse the language and the grammatical rule is $S \rightarrow SS$. At the second stage of parsing, two processors apply the same rule $S \rightarrow SS$ on the two nonterminals "S" and the parsing result is SSSS. At the third stage of parsing, four processors apply the same rule $S \rightarrow a$ on all the four nonterminals "S". Hence, the parsing result is aaaa. This language cannot be passed sequentially as the language is defined to be parsed only parallelly. The sentence aaaa of this example is parsed by the parallel context-free grammar G. Thus, the sentence is accepted as a member of $L(G)$.

In the preprocessing part of the tree grammar approach in Chapter 4, the task of horizontal and vertical processing is assigned a flag M1 which denotes the MIMD mode for the task. The horizontal and vertical

processing task is thus allocated to the parallel processor (PP) of the CAIP by the tag examination routine of the Task Management Processor (TMP). The control unit of the parallel processor (PP) loads the operating system of the MIMD mode after finding the HI flag of the task. The parallel processor has N processors. Half of the set of N processors is designated to execute the horizontal processing and the other half of that set of processors executes the vertical processing. The different instructions are executed on multiple picture data at the same time by the parallel processor (PP). In this example, the computer architecture CAIP utilizes the multiple instruction stream multiple data stream (MIMD) mode to achieve high system throughput for this task.

5.4 SUMMARY AND REMARKS

A computer architecture for image processing (CAIP) has been proposed. This computer architecture is designed by the distributed computing approach. This computer is comprised of a parallel processor (PP) and a sequential arithmetic processor (SAP). The flexibility and high performance of this computer architecture are contributed to by two major features which are the reconfiguration capability, described in section 5.2.1, and the method of computer exploitation of task parallelism, stated in section 5.2.2. This computer architecture for image processing is proposed to use microprocessors as the processing elements. The advantage of using a microprocessor array is that the cost of microprocessors is much lower than that of conventional processors. The disadvantage is that the processing power of microprocessors is less than that of conventional processors, especially in addressing capability. For example, the most popular microprocessors INTEL 8080 and MOTOROLA 6800 do not have associate addressing or microprogramming abilities.

For special image processing computers, some processing powers of conventional processors, such as associative addressing, are not essential [102]. The approach of designing a general purpose computer by microprocessors has been controversial. But, in designing a special purpose computer for image processing, microprocessors have their advantages. Furthermore, the advance in semiconductor technology is toward the development of microprocessors with higher processing power. The recent developments in the microprocessors services INTEL 3000, MOTOROLA M2900, and Texas Instruments 74S481 have provided microprogramming capability to microprocessors.

With the fast growth of image processing and its applications, the need for a special image processing machine such as the proposed computer, CAIP, should certainly be appreciated.

CHAPTER 6

CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

6.1 SUMMARY AND CONCLUSIONS

In this study, we have presented the finite-state string grammar approach for image recognition, a tree grammar approach for image segmentation, and a special computer architecture for image processing.

The system for image recognition was presented in Chapter 3. This system consists of preprocessor, syntactic analyzer, and postprocessor [118]. Two methods were presented; namely, syntax-directed [7] and syntax-controlled methods. The finite-state string grammar was applied to the recognition of highways and rivers from LANDSAT images. For the syntax-controlled method, the finite-state string grammar was automatically inferred by the k-tail finite-state grammatical inference procedure. From the experiments, the finite-state string grammar which is inferred by the $k=2$ case in the grammatical inference procedure was found to be the most suitable of those investigated for highway and river recognition. Some further applications of the syntactic method were bridge and commercial/industrial area recognition. The method extracted the structural and contextual information from the images to recognize the objects of interest. The locations and lengths of bridges, as well as the centers and sizes of commercial/industrial areas were extracted by the appropriate algorithms described in Chapter 3.

In Chapter 4, a syntactic method for image segmentation was presented. This method is a tree grammar approach which utilizes a tree automaton to extract the boundaries of the homogeneous region segments of an image. The homogeneity of the region segment was obtained through the texture feature measurements of the image. The experiments were conducted on different images obtained from satellite, and infrared sensors. The results of syntactic image segmentation compare favorably with those of statistical classification techniques.

In Chapter 5, a computer architecture for image processing was proposed. This computer architecture was designed by the distributed computing approach. This computer consists of a parallel processor and a sequential arithmetic processor. Two major features which are new to the field of special purpose computer architectures for image processing contributed to the flexibility and high performance of this architecture. These features are the reconfigurable capability, described in section 5.2.1, and the method of computer exploitation of task parallelism, given in section 5.2.2.

In conclusion, the syntax-controlled method for image recognition was found to be more powerful than the syntax-directed method, commonly known as template matching. Firstly, the computer processing of the recognition process for the syntax-controlled method is faster than that of the syntax-directed method. Secondly, the recognition by the syntax-controlled method is based on an automaton (or parser) which is much more powerful and flexible than recognition by matching the templates in the syntax-directed method. Also the recognition power of the syntax-directed method is limited by the number of templates. The advantage of the syntax-directed method is its fast software development time

which means the time of developing and implementing such a method is fast and less complicated than the syntax-controlled method. However, once the software of the syntax-controlled method is developed, the syntax-controlled method saves program execution time in performing the task every time.

The selection of an appropriate grammatical approach for recognition usually depends on the problem requirements [2]. The string grammar is most suitable in describing a string-like pattern, such as, highways and rivers. Thus, the string grammar approach, which uses a finite-state automaton as syntactic analyzer, is applied for the image recognition in Chapter 3.

For the problems in Chapter 4, the patterns of interest are the boundaries of the image segments. These boundary patterns are high dimensional. The tree grammar is more convenient in describing high dimensional objects than string grammar, therefore, a tree grammar was used. The tree grammar offers a natural high dimensional generalization of strings and the tree automaton has a high analytical capability in recognizing patterns. Thus, the tree grammar approach, which utilizes a tree automaton as syntactic analyzer, is applied for image segmentation in Chapter 4.

We believe that the syntactic algorithms for image recognition and segmentation developed in Chapter 3 and 4 provide a better way to understand image structure and to extract image information than these have previously been done. These syntactic algorithms can be used for military reconnaissance, industrial automation, and medical diagnosis.

In addition, the fast growth of image processing and its impact on industrial, biomedical, and military applications, has created a need

for a special image processing computer such as that proposed in Chapter 5, namely, the computer, CAIP.

6.2 SUGGESTIONS FOR FURTHER WORK

Syntactic algorithms and special computer architectures for image processing are very important areas of research. There are still many open problems that are worth investigating.

The finite-state string grammar and the tree grammar approaches need to be further developed for more and varied applications. Other grammatical approaches, such as a context-free grammar [2] and a parallel context-free grammar [103] need investigation for their applications to image processing. Parallel parsing schemes for finite-state, context-free, and context-sensitive grammars need to be studied. And the special computer described in Chapter 5 needs to be physically constructed according to the proposed computer architecture in Chapter 5, and put into operation.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. K.S. Fu, "On the application of pattern recognition techniques to remote sensing problems," TR-EE 71-13, Purdue University, Indiana, June 1971.
2. K.S. Fu, Syntactic Methods in Pattern Recognition, Academic Press, New York, 1974.
3. K.S. Fu, "Pattern recognition in remotely sensing of the earth's resources," IEEE Trans. on Geoscience Electronics, Vol. GE-14, pp. 10-18, January 1976.
4. K.S. Fu, ed., Syntactic Pattern Recognition, Applications, Springer-Verlag, Communication and Cybernetics, Vol. 14, 1977.
5. K.S. Fu, "Syntactic (Linguistic) pattern recognition," in Digital Pattern Recognition, ed. by K.S. Fu, Springer-Verlag, Communication and Cybernetics, Vol. 10, 1976.
6. K.S. Fu and A. Rosenfeld, "Pattern recognition and image processing," IEEE Trans. on Computers, Vol. C-25, Dec. 1976.
7. Janmin Keng and K.S. Fu, "A syntax-directed method for land-use classification of LANDSAT images," Proceedings of the Symposium on Current Mathematical Problems in Image Science, Monterey, California, pp. 261-265, Nov. 10-12, 1976.
8. Janmin Keng, "Image segmentation and object detection by a syntactic method," Proceedings of Workshop on Image Understanding, Advanced Research Projects Agency, Minneapolis, Minnesota, pp. 38-43, April 20, 1977.
9. Janmin Keng, "A syntactic method for image segmentation," Proceedings of Seventh Annual Symposium of Automatic Imagery Pattern Recognition, Electronic Industrial Association, College Park, Maryland, pp. 62-87, May 23-24, 1977.
10. K.S. Fu and Janmin Keng, "A model of automatic information extraction for image understanding," Quarterly Progress Report, Aug. 1, 1976 - Oct. 31, 1977, Research on Image Understanding and Information Extraction, School of Electrical Engineering, Purdue University, W. Lafayette, Indiana.

11. B. Moeyer and K.S. Fu, "A tree system approach for fingerprint pattern recognition," IEEE Trans. on Computers, Vol. C-25, pp. 262-275, March 1976.
12. K.S. Fu, "Syntactic methods and image processing," Proceedings of Conference on Pattern Recognition and Image Processing, Troy, New York, pp. 19-20, June 6-8, 1977.
13. K.S. Fu and B.K. Bhargava, "Tree systems for syntactic pattern recognition," IEEE Trans. on Computers, Vol. C-22, No. 12, pp. 1087-1099, Dec. 1973.
14. J.M. Brayer and K.S. Fu, "Web grammar and its application to pattern recognition," Ph.D. Thesis, School of Electrical Engineering, Purdue University, W. Lafayette, Indiana, 1975.
15. R.Y. Li and K.S. Fu, "Tree system approach for LANDSAT data interpretation," Proceedings of Symposium of Machine Processing of Remotely Sensed Data, Purdue University, W. Lafayette, Indiana, pp. 2A-10-2A-17, 1976.
16. S.Y. Lu and K.S. Fu, "Structure-preserved error-correcting tree automata for syntactic pattern recognition," Proceedings of Conference of Pattern Recognition and Image Processing, Troy, New York, June 6-8, 1977.
17. K.S. Fu and S.Y. Lu, "A clustering procedure for syntactic patterns," Proceedings Workshop on Picture Data Description and Management, Chicago, Illinois, April 21-22, 1977.
18. K.S. Fu and P.H. Swain, "On syntactic pattern recognition," in Software Engineering, Vol. 2, ed. by J.T. Tou, Academic Press, 1970.
19. L.M. Fung and K.S. Fu, "Stochastic syntactic decoding for pattern classification," IEEE Trans. on Computers, Vol. C-24, No. 6, July 1975.
20. K.S. Fu and T.L. Booth, "Grammatical inference: Introduction and survey - Part I and II," IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-5, 1975.
21. B.K. Bhargava and K.S. Fu, "Transformations and inference of tree grammars for syntactic pattern recognition," Proceedings of IEEE Int. Conference Systems, Man and Cybernetics, Dallas, Texas, Oct. 2-4, 1974.
22. K.S. Fu, "Error-correcting parsing for syntactic pattern recognition," Data Structure, Computer Graphic and Pattern Recognition, ed. by A. Klinger, et. al., Academic Press, 1977.
23. T. Pavlidis, "Syntactic feature extraction for shape recognition," Technical Report No. 213, Princeton University, April 1976.

24. R.L. Kettig and D.A. Landgrebe, "Automatic boundary and sample classification of remotely sensed multispectral data," LARS Information Note 041773, W. Lafayette, Indiana.
25. T. Huang, "Per field classifier for agricultural application," LARS Information Note 060569, Purdue University, W. Lafayette, Indiana.
26. S.W. Zucker, A. Rosenfeld and L.S. Davis, "Picture segmentation by texture discrimination," IEEE Trans. on Computers, Vol. 24, pp. 1228-1233, Dec. 1975.
27. R.J. Wall, A. Klinger and K.R. Castleman, "Analysis of image histograms," Second International Joint Conference on Pattern Recognition, pp. 341-344, 1974.
28. R.J. Wall, "The grey level histogram for threshold boundary determination in image processing with application to the scene segmentation problem in human chromosome analysis," Ph.D. Dissertation, UCLA, 1974.
29. L.S. Davis, A. Rosenfeld and J.S. Weszka, "Region extraction by averaging and thresholding," IEEE Trans. on Systems, Man and Cybernetics, pp. 383-388, May 1975.
30. J.S. Weszka, R.N. Nagel and A. Rosenfeld, "A threshold selection technique," IEEE Trans. on Computers, Vol. 23, pp. 1322-1326, Dec. 1974.
31. F. Tomita, M. Yachida and S. Tsuji, "Detection of homogeneous regions by structural analysis," in Proceedings Third International Joint Conference on Artificial Intelligence, pp. 564-571, 1973.
32. S. Tsuji and F. Tomita, "A structural analyzer for a class of textures," Computer Graphics and Image Processing 2, pp. 216-231, 1973.
33. M. Yachida and S. Tsuji, "Application of color information to visual perception," Pattern Recognition 3, pp. 307-323, 1971.
34. C.R. Brice and C.L. Fennema, "Scene analysis using regions," Artificial Intelligence Journal, Vol. 1, No. 3, Fall 1970.
35. T.P. Strong and A. Rosenfeld, "A region coloring technique for scene analysis," Com. ACH, Vol. 6, pp. 237-246, April 1973.
36. C.A. Harlow and S.A. Eisenbeis, "The analysis of radiographic images," IEEE Trans. on Computers, Vol. C-22, pp. 678-689, July 1973.
37. E.M. Rodd, "Closed boundary field selection in multispectral digital images," IBM Publication No. 320, 2420, January 1972.

38. R.L. Kettig and D.A. Landgrebe, "Computer classification of remotely sensed multispectral image data by extraction and classification of homogeneous objects," Ph.D. Thesis, School of Electrical Engineering, Purdue University, W. Lafayette, Indiana, May 1975.
39. R. Bajcsy, "Computer identification of visual surfaces," Computer Graphics and Image Processing, pp. 118-130, No. 2, 1973.
40. J.N. Gupta and P.A. Wintz, "Computer processing algorithms for locating boundaries in digital picture," 1974 International Joint Conference on Pattern Recognition, pp. 155-156, 1974.
41. S. Tsuji and R. Fujiwara, "Linguistic segmentation of scenes into regions," 1974 International Joint Conference on Pattern Recognition, pp. 104-108, 1974.
42. Y. Yakimovsky and J. Feldman, "A semantics-based decision theory region analyzer," Proceedings Third International Joint Conference on Artificial Intelligence, Stanford, pp. 580-588, Aug. 1973.
43. Y. Yakomovsky, "Picture processing: use of probabilistic semantics for region growing and recognition," Ph.D. Dissertation, Stanford University, California, 1973.
44. T.V. Robertson, K.S. Fu and P.H. Swain, "Multispectral image partitioning," Ph.D. Thesis, School of Electrical Engineering, Purdue University, W. Lafayette, Indiana, Aug. 1973.
45. A. Klinger, "Data structures and pattern recognition," Proceedings First International Joint Conference on Pattern Recognition, pp. 497-498, Oct. 1973.
46. S.L. Horowitz and T. Pavlidis, "Picture segmentation by a directed split and merge procedure," Proceedings International Joint Conference on Pattern Recognition, pp. 424-433, 1974.
47. A.K. Griffith, "Computer recognition of prismatic solids," Proj. MAC Technical Report, MAC-TR-73, MIT, Cambridge, Mass., Aug. 1970.
48. A.K. Griffith, "Edge detection in simple scene using a priori information," IEEE Trans. Computers, Vol. C-22, pp. 371-381, 1973.
49. A.K. Griffith, "Mathematical models for automatic line detection," Communications ACM, Vol. 20, No. 1, pp. 62-80, 1973.
50. M.H. Hueckel, "An operator which locates edges in digitized picture," Journal of ACM, Vol. 18, No. 1, pp. 113-125, January 1973.
51. M.H. Hueckel, "A local visual operator which recognizes edges and lines," Journal of ACM, Vol. 20, No. 4, pp. 634-647, Oct. 1973.
52. A. Rosenfeld, "Connectivity in digital pictures," Journal of ACM, Vol. 17, No. 1, pp. 146-160, Jan. 1970.

53. M.D. Kelley, "Edge detection in picture by computer using planning," in Machine Intelligence 6, pp. 397-409, American Elsevier Publ., Co., New York, 1970.
54. P.V.C. Hough, "Method and means for recognizing complex patterns," U.S. Patent 3.069.654, Dec. 18, 1962.
55. R.O. Duda and P.E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," Comm. ACM, Vol. 15, pp. 11-15, Jan. 1972.
56. T. Pavlidis, "Segmentation of pictures and maps through functional approximation," Computer Graphics and Image Processing, No. 1, pp. 360-372, 1972.
57. R.O. Duda and P.E. Hart, "Pattern classification and scene analysis," Wiley-Interscience Publication, New York, 1973.
58. A. Rosenfeld and M. Thurston, "Edge and curve detection for visual scene analysis," IEEE Trans. Computers, Vol. C-20, pp. 562-569, May 1971.
59. A. Rosenfeld, M. Thurston and Y.H. Lee, "Edge and curve detection: Further experiments," IEEE Trans. Computers, Vol. C-21, No. 7, July 1972.
60. A. Rosenfeld, R.B. Thomas and Y.H. Lee, "Edge and curve detection for texture discrimination," in Picture Processing and Psychopictories, B.S. Lapkin and A. Rosenfeld, Eds., Academic Press, New York, pp. 381-393, 1970.
61. A. Rosenfeld, "A nonlinear edge detection technique," Proc. IEEE Trans., Vol. 58, pp. 814-816, May 1970.
62. K.K. Pingle, "Visual perception by a computer automatic interpretation and classification of images," A. Grasselli (ed.) Academic Press, New York, pp. 277-284, 1969.
63. P. E. Anuta, "Spatial registration of multispectral and multitemporal imagery using fast Fourier transformation techniques," IEEE Trans. Geoscience Electronics, Vol. GE-8, No. 4, pp. 353-368, Oct. 1970.
64. J.M. Prewitt, "Object enhancement and extraction," Picture Processing and Psychopictories, B.S. Lapkin and A. Rosenfeld (eds.), Academic Press, New York, pp. 113-114, 1970.
65. A.G. Wacker and D.A. Landgrebe, "Boundaries in multispectral imagery by clustering," IEEE Symposium on Adaptive Processes, Decision and Control (9th), University of Texas (Austin), Dec. 1970.
66. A. Rosenfeld and E.B. Troy, "Visual texture analysis," Conference Rec., 1970 Symposium Feature Extraction and Selection in Pattern Recognition, IEEE Publ. 70C-51C, pp. 115-124, Oct. 1970.

67. E. Persoon, "A new edge detection algorithm and its applications in picture processing," TR-EE 75-38, Purdue University, W. Lafayette, Indiana, Oct. 1975.
68. R.B. Ohlander, "Analysis of natural scenes," Ph.D. Dissertation, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, April 1975.
69. A. Konger and C.R. Dyer, "Experiments on picture representation using regular decomposition," Computer Graphics and Image Processing, pp. 68-105, March 1976.
70. S.L. Horowitz and T. Pavlidis, "Picture processing by graph analysis," Proceedings of the Conference on Computer Graphics, Pattern Recognition and Data Structure, pp. 125-129, May 1975.
71. S.L. Horowitz and T. Pavlidis, "Picture segmentation by a tree traversal algorithm," Journal of the Association for Computing Machinery, pp. 368-388, April 1976.
72. R.M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-3, pp. 610-621, Nov. 1973.
73. J.S. Weszka, C.R. Dyer and A. Rosenfeld, "A comparative study of texture measures for terrain classification," IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-6, No. 4, pp. 269-285, April 1976.
74. R. Bajcsy, "Computer description of textured surfaces," Int. Proc., Third Int. Joint Conference on Artificial Intelligence, pp. 572-579, Aug. 1973.
75. S.W. Zucker, "Toward a model of texture," Computer Graphics and Image Processing, No. 5, pp. 190-202, June 1976.
76. A. Rosenfeld and B. Lipkin, "Texture synthesis in picture processing and psychopictorics," Academic Press, New York, 1970.
77. M.M. Galloway, "Texture classification using gray level runlengths," Computer Graphics and Image Processing, Vol. 4, pp. 172-179.
78. A. Rosenfeld, R. Hummel, and S.W. Zucker, "Scene labeling by relaxation operations," IEEE Trans. on Systems, Man, and Cybernetics, SMC-6, pp. 420-433, 1976.
79. J.M. Tenenbaum and H.G. Barrow, "IGS: A paradigm for integrating image segmentation and interpretation," Proc. 3IJCPR, pp. 504-513, Nov. 1976.
80. G.J. Vanderbrug, "Experiments in interactive enhancement of linear features," Computer Graphics and Image Processing, pp. 25-42, 1977.

81. R. Bajcsy and M. Tavakoli, "A computer recognition of bridges, islands, rivers and lakes from satellite pictures," Proceedings of Symposium of Machine Processing of Remotely Sensed Data, Purdue University, W. Lafayette, Indiana, pp. 2A-55 to 2A-68, Oct. 1973.
82. J.J. Horan, "LANDSAT" Multispectral eye in the sky," IEEE Spectrum, Vol. 13, No. 3, pp. 56-62, March 1976.
83. A. Rosenfeld and A.C. Kak, Digital Picture Processing, Academic Press, New York, 1976.
84. W.J. Todd and M.F. Baumgardner, "Land-use classification of Marion County, Indiana, by spectral analysis and digitized satellite data," LARS Information Note 101673, LARS, Purdue University, 1973.
85. W.B. Thompson, "Textural boundary analysis," IEEE Trans. Computers Vol. C-26, pp. 272-276, March 1977.
86. S.G. Carlton and O.R. Mitchell, "Image segmentation using texture and gray level," Proc. Conference Pattern Recognition and Image Processing, Troy, New York, pp. 387-391, June 6-8, 1977.
87. B.H. McCormick, "The Illinois pattern recognition computer-ILLIAC III," IEEE Trans. on Electronic Computers, EC-12, No. 5, pp. 791-813, 1963.
88. B. Kruse, "A parallel picture processing machine," IEEE Trans. on Computers, Vol. C-22, No. 12, pp. 1075-1087, Dec. 1973.
89. C.D. Stamopoulos, "Parallel algorithms for joining two points by a straight-line segment," IEEE Trans. on Computers, pp. 642-646, June 1974.
90. C.D. Stamopoulos, "Parallel image processing," IEEE Trans. on Computers, Vol. C-24, No. 4, pp. 424-433, April 1975.
91. "CLIP 3 operating manual," Image Processing Group, Dept. Phys. Astron. University College London, England, Rep. 73/4.
92. M.J.B. Duff, "CLIP4: A large scale integrated circuit array parallel processor," Int. Joint Conference Pattern Recognition, Coronado, California, Nov. 1976.
93. H. Shinada, H. Asada, T. Kondo, K. Mori, M. Kidode and H. Numagami, "An interactive image processing system - TOSPICS and its application to remote sensing," Proc. of Seventh Annual Symposium of Automatic Imagery Pattern Recognition, College Park, Maryland, pp. 177-183, May 23-24, 1977.
94. E.G. Coffman, Jr., G.J. Barnett and R.A. Snowdon, "On the performance of interleaved memories with multiple word bandwidth," IEEE Trans. on Computers, Vol. C-20, pp. 1570-1573, Dec. 1971.

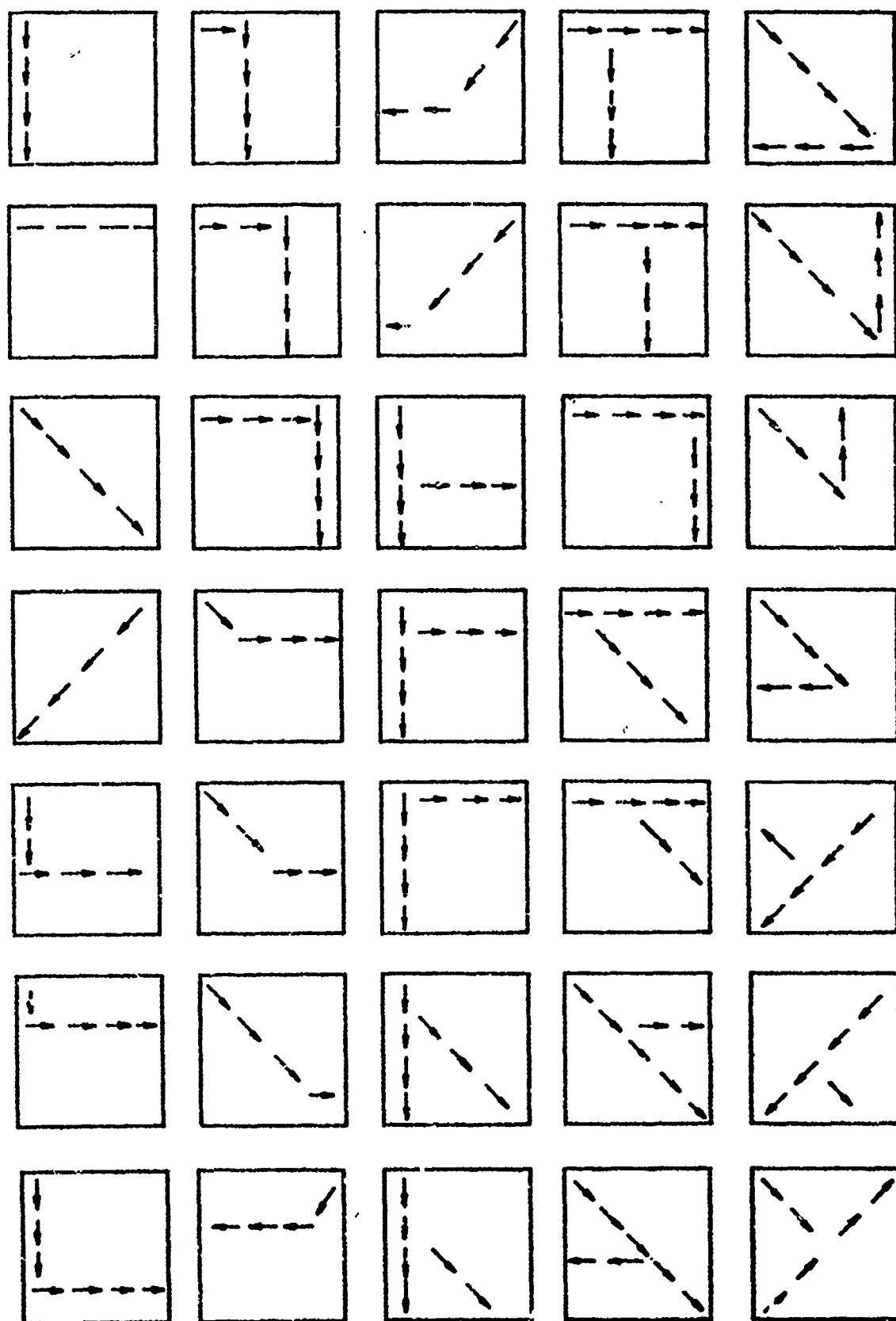
95. W.S. Brainerd, "Tree generating regular systems," *Information and Control*, pp. 217-231, 1969.
96. T. Pavlidis, "The application of a general pattern recognition system to the automatic inspection and description of printed wiring board," *Proc. Seventh Annual Symposium of Automatic Imagery Pattern Recognition*, College Park, Maryland, May 23-24, 1977.
97. G.R. Allen, L.O. Bonrud, J.J. Cosgrove and R.M. Stone, "The design and use of special purpose processors for the machine processing of remotely sensed data," *Machine Processing of Remotely Sensed Data*, LARS, Purdue University, 1973.
98. L. Cordello, M.J.B. Duff and S. Levialdi, "Comparing sequential and parallel processing of pictures," *Third Int. Joint Conference Pattern Recognition*, Coronado, California, Nov. 1976.
99. B. Kruse, "The PICAP picture processing laboratory," *Third Int. Joint Conference on Pattern Recognition*, Coronado, California, Nov. 1976.
100. Ramsey, ed., Image Processing in Biological Science, University of California Press, Berkeley-Los Angeles, 1968.
101. E. Persoon and K.S. Fu, "A minicomputer facility for picture processing and pattern recognition research," *Computer*, Vol. 9, No. 5, pp. 70-78, May 1976.
102. K.J. Thurber and L.D. Wald, "Associative and parallel processor," *Computer Surveys*, Vol. 7, No. 4, pp. 215-255, Dec. 1975.
103. R. Siromoney and K. Krithivasan, "Parallel context free languages," *Information and Control*, Vol. 24, pp. 155-162, 1974.
104. D.L. Slotnik, "Unconventional Systems," *Proceedings 1967 Spring Joint Computer Conference*, pp. 477-481; G.H. Barnes, et al, "The ILLIAC computer," *IEEE Trans. on Computers*, Vol. C-17, pp. 746-757, 1968.
105. T.C. Chen, "The overlap design of the IBM system/360 model 92 control processing unit," *Proceedings 1964 Fall Joint Computer Conference*, Part II, pp. 73-80, 1964.
106. H.S. Stone, "Program chairman's remark," *Proceedings of Fourth Annual Symposium on Computer Architecture*, March 23-25, 1977.
107. A.E. Akers, "An auxiliary memory controller for array processing," *Technical Report 76-3*, School of Electrical Engineering, Purdue University, W. Lafayette, Indiana.
108. P.H. Enslow, Jr., ed., Multiprocessors and Parallel Processing, John Wiley & Sons, Inc., 1974.

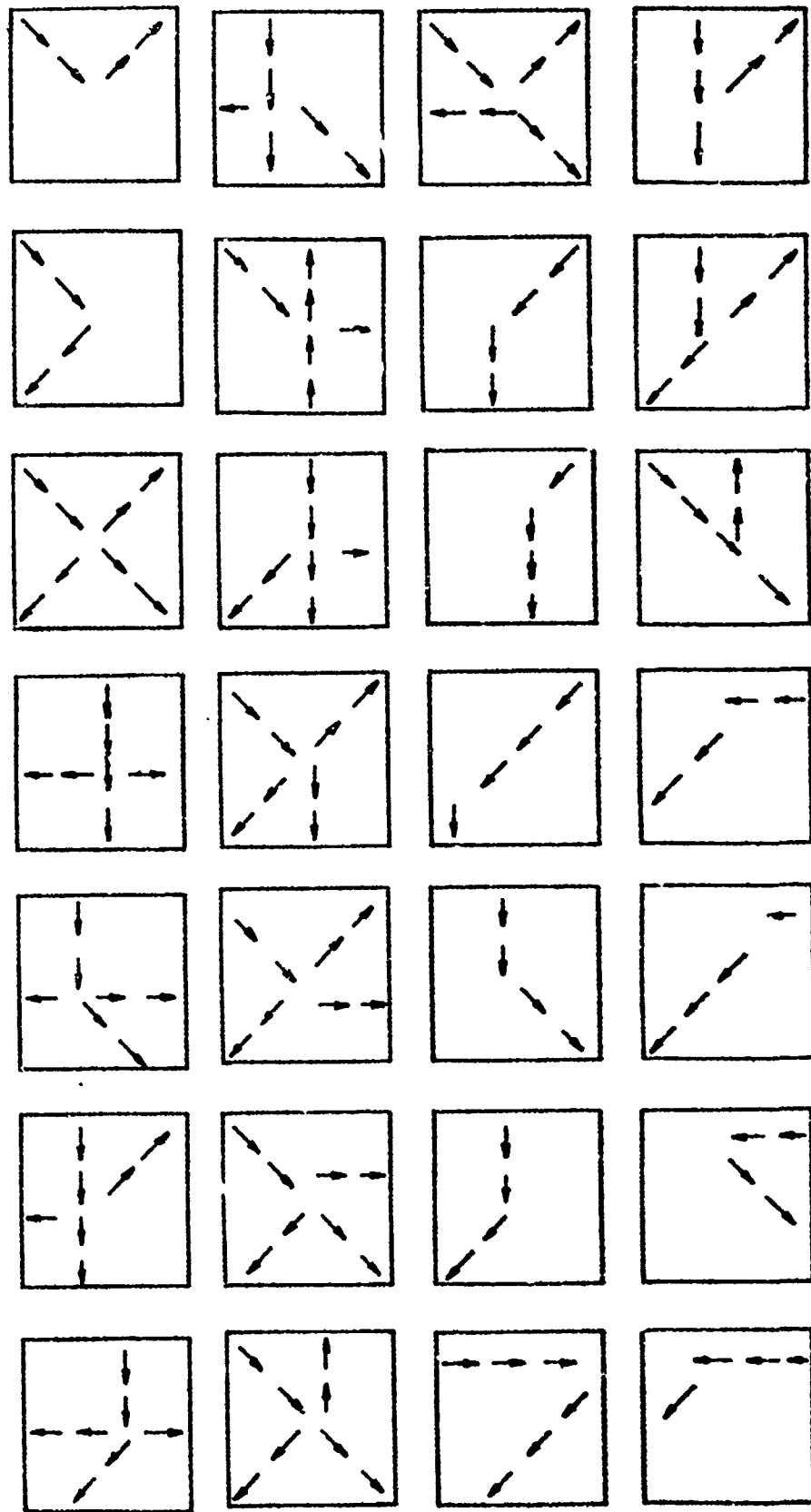
109. J.M. Vocar and R.O. Faiss, "Warp processing using STARAN," Proc. of Workshop on Picture Data Description and Management, Chicago, Illinois, April 21-22, 1977.
110. A.V. Aho and J.D. Ullman, The Theory of Parsing, Translation, and Compiling, Volume 1: Parsing, Prentice-Hall, 1972.
111. H.S. Stone, ed., Introduction to Computer Architecture, Science Research Associates, Inc., 1975.
112. H. Wechsler, "Automatic detection of rib contours in chest radiographs," Ph.D. Thesis, University of California - Irvine, Irvine, California, 1976.
113. D.E. Knuth, The Art of Computer Programming, Volume 1 - Fundamental, Addison-Wesley, 1968.
114. R.A. Jarvis, "Region based image segmentation using shared near neighbor clustering," Seventh International Conference on Cybernetics and Society, Washington, D.C., September 19-21, 1977.
115. Image Processing System (TOSPICS) Report, April 1976, Tokyo Shibaura Electric Co. LTD., Computer Division.
116. K. Mori, H. Shinoda and H. Asada, "Toshiba pattern information cognition system - TOSPICS," Toshiba Review, No. 107, Jan. 1977.
117. S.Y. Lu and K.S. Fu, "A syntactic approach to texture analysis," Technical Report TR-EE 77-14, Purdue University, 1977.
118. A Proposal to Defense Advanced Research Projects Agency on Automatic Image Recognition System, Honeywell, Inc., Systems and Research Center, Minneapolis, Minnesota, March 1977.
119. Janmin Keng and K.S. Fu, "A system of computerized automatic pattern recognition for remote sensing," Proceedings of International Computer Symposium, Taipei, Rep. of China, Dec. 27-29, 1977.

APPENDICES

APPENDIX A

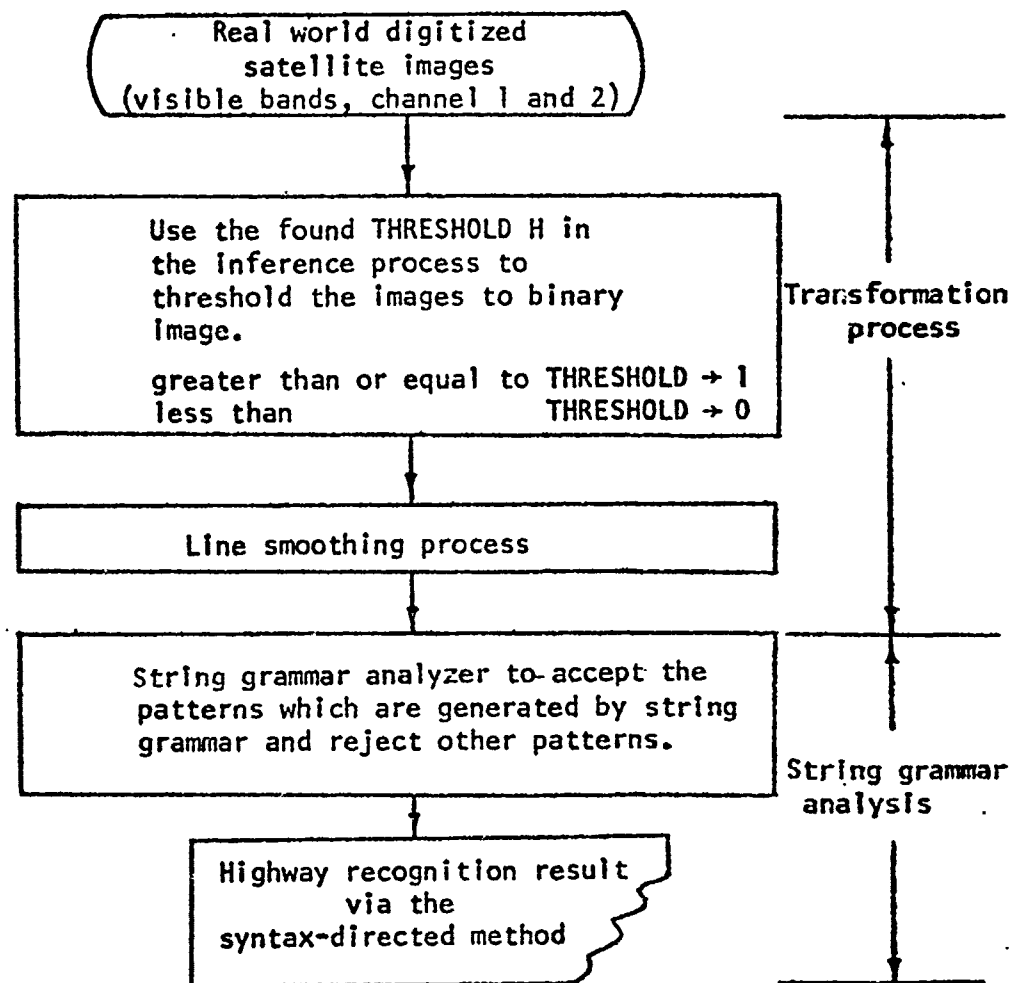
SAMPLE PATTERNS FOR INFERENCE OF THE
TREE GRAMMAR FOR IMAGE SEGMENTATION





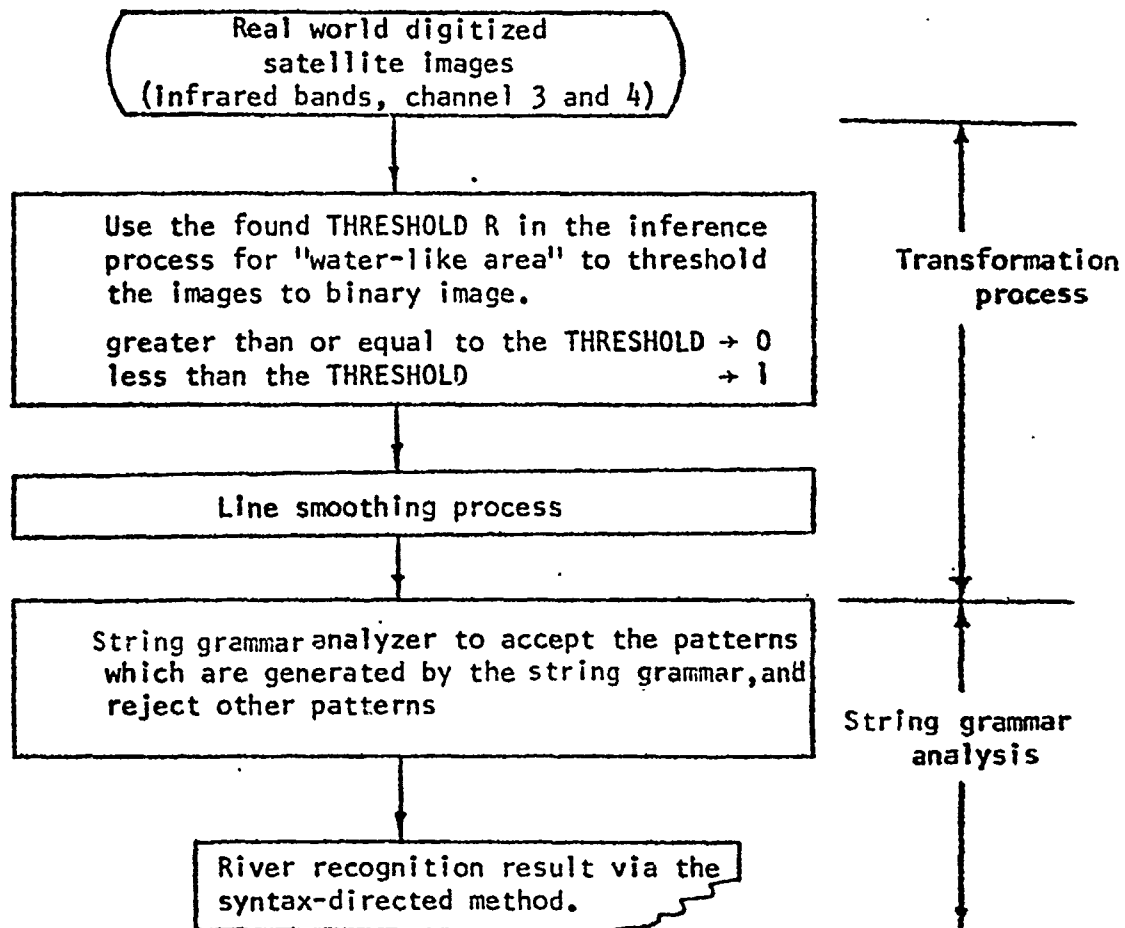
APPENDIX B

THE FLOW CHART OF HIGHWAY RECOGNITION VIA
THE SYNTAX-DIRECTED METHOD



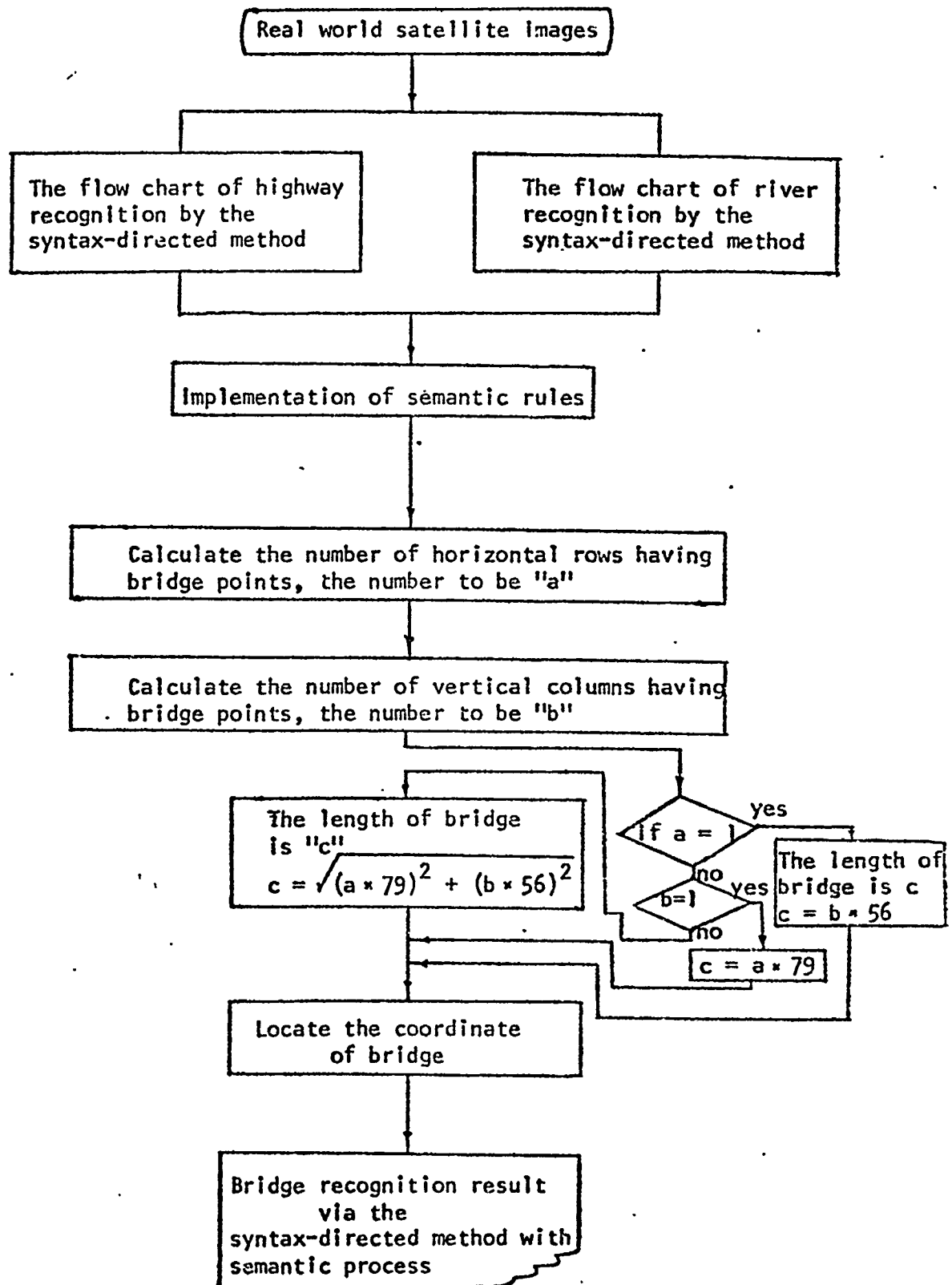
APPENDIX C

THE FLOW CHART OF RIVER RECOGNITION VIA
THE SYNTAX-DIRECTED METHOD



APPENDIX D

THE FLOW CHART OF BRIDGE DETECTION VIA THE SYNTAX-
DIRECTED METHOD WITH SEMANTIC PROCESS



APPENDIX E

TRAINING SAMPLES FOR HIGHWAY GRAMMAR

BEST AVAILABLE COPY

SAMPLE PATTERNS

1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0
0 0 0 0 1 0 0 0
0 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0
0 1 0 0 0 0 0 0
1 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0

1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0
1 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

BEST AVAILABLE COPY

1 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0
 0 0 1 1 1 1 1 1
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

1 1 1 1 1 0 0 0
 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

0 1 1 1 1 1 1 1
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1
 0 0 0 0 0 0 1 0
 0 0 0 0 0 1 0 0
 0 0 0 0 1 0 0 0
 1 1 1 1 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0
 0 0 1 0 0 0 0 0
 0 0 0 1 0 0 0 0
 0 0 0 0 1 1 1 1
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

0 0 0 1 1 1 1 1
 0 0 1 0 0 0 0 0
 0 1 0 0 0 0 0 0
 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

BEST AVAILABLE COPY

1 1 1 0 0 0 0 0
 0 0 0 1 0 0 0 0
 0 0 0 0 1 0 0 0
 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0
 0 0 1 0 0 0 0 0
 0 0 0 1 0 0 0 0
 0 0 0 0 1 0 0 0
 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 1
 0 0 0 0 0 0 1 0
 0 0 0 0 0 1 0 0
 0 0 0 0 1 0 0 0
 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 0
 1 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

0 0 0 0 0 1 1 1
 0 0 0 0 1 0 0 0
 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 0
 0 1 0 0 0 0 0 0
 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0
 0 0 1 1 1 1 1 1
 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1
 0 0 0 0 0 0 1 0
 0 1 1 1 1 1 0 0
 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

BEST AVAILABLE COPY

1 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0
 0 0 1 0 0 0 0 0
 0 0 0 1 0 0 0 0
 0 0 0 0 1 1 1 0
 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0

0 0 0 1 1 1 1 1
 0 0 1 0 0 0 0 0
 1 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1
 0 0 0 0 0 0 1 0
 0 0 0 0 0 1 0 0
 0 0 0 0 1 0 0 0
 0 1 1 1 0 0 0 0
 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0
 0 0 1 0 0 0 0 0
 0 0 0 1 0 0 0 0
 0 0 0 0 1 0 0 0
 0 0 0 0 1 0 0 0
 0 0 0 0 1 0 0 0
 0 0 0 0 1 0 0 0

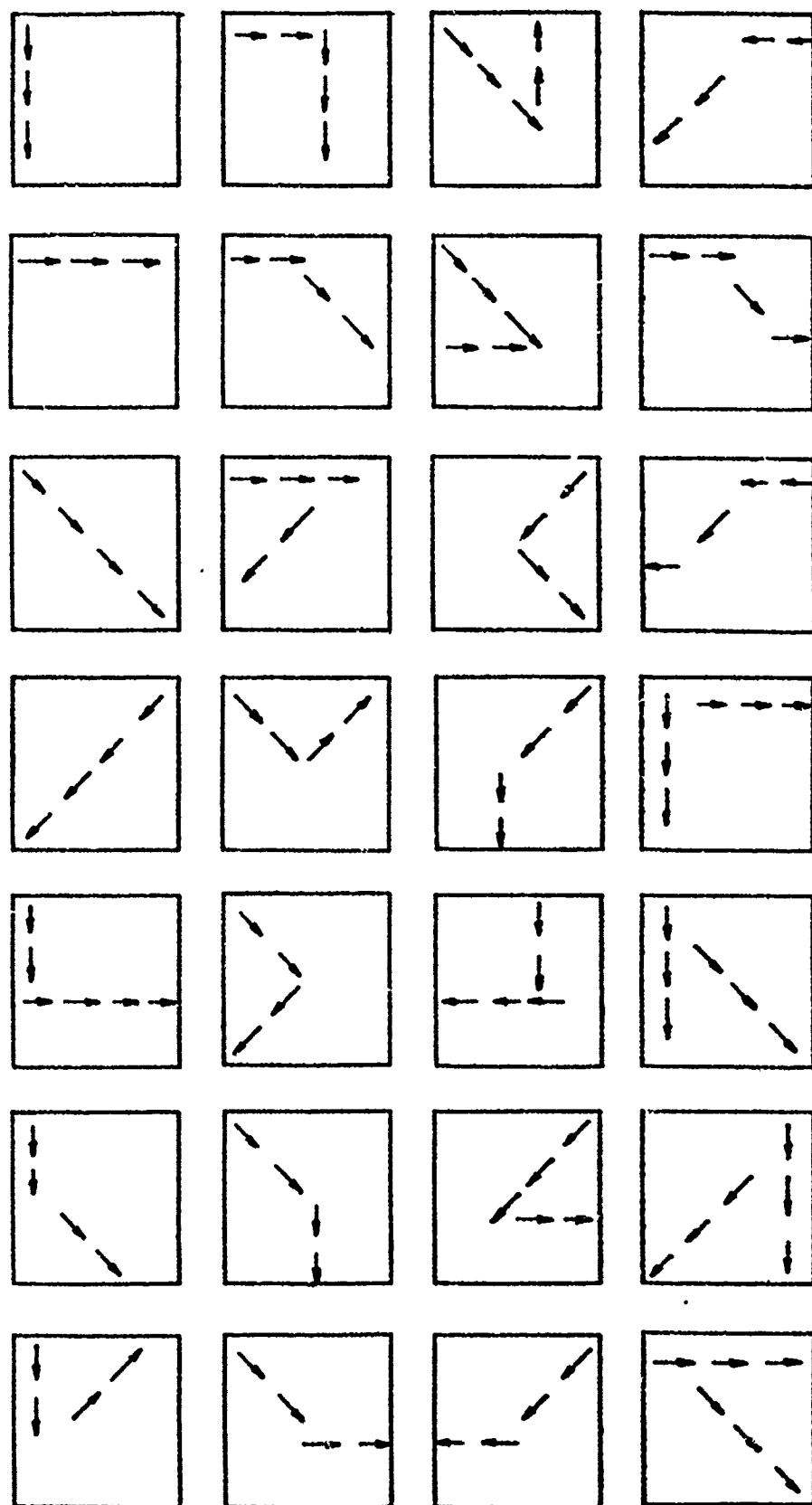
1 1 1 1 1 0 0 0
 0 0 0 0 0 1 0 0
 0 0 0 0 0 1 1 1
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0

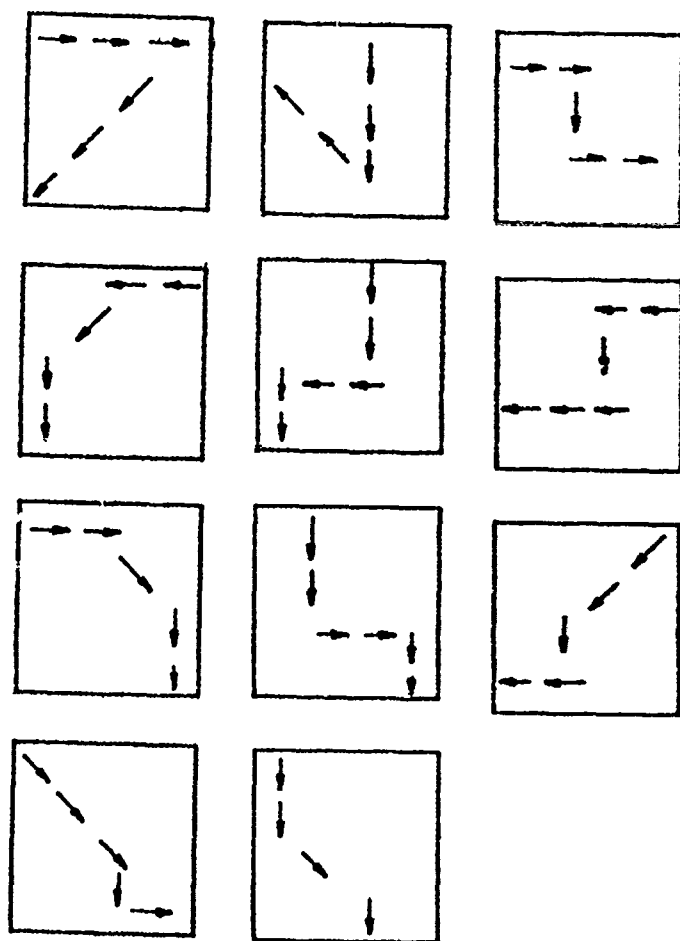
0 0 0 0 0 0 0 1
 0 0 0 0 0 0 1 0
 0 0 0 0 0 1 0 0
 0 0 0 0 1 0 0 0
 0 0 0 0 1 0 0 0
 0 0 0 0 1 0 0 0
 0 0 0 0 1 0 0 0
 0 0 0 0 1 0 0 0

APPENDIX F

SAMPLE PATTERNS FOR INFERENCE OF TREE

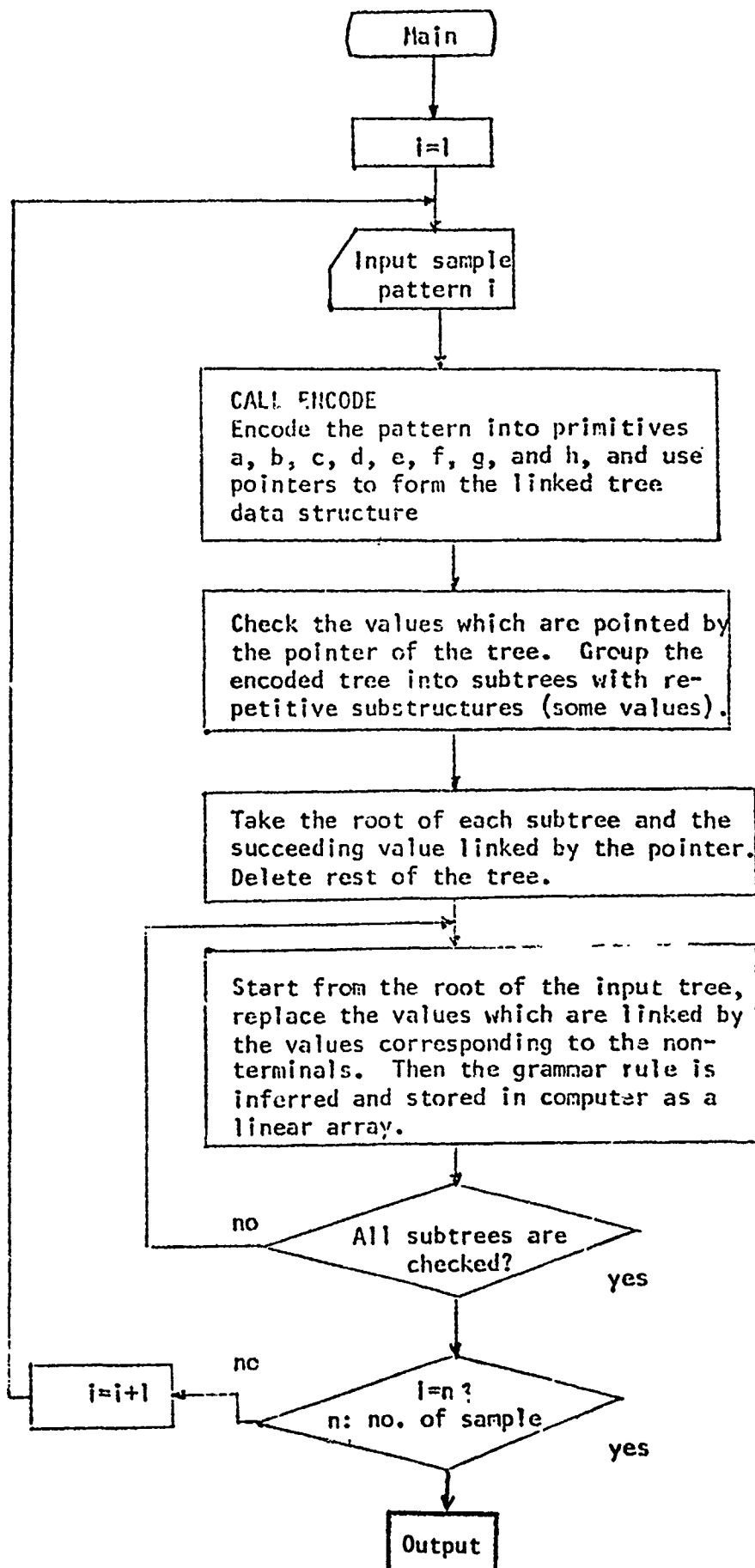
GRAMMAR FOR MILITARY VEHICLE





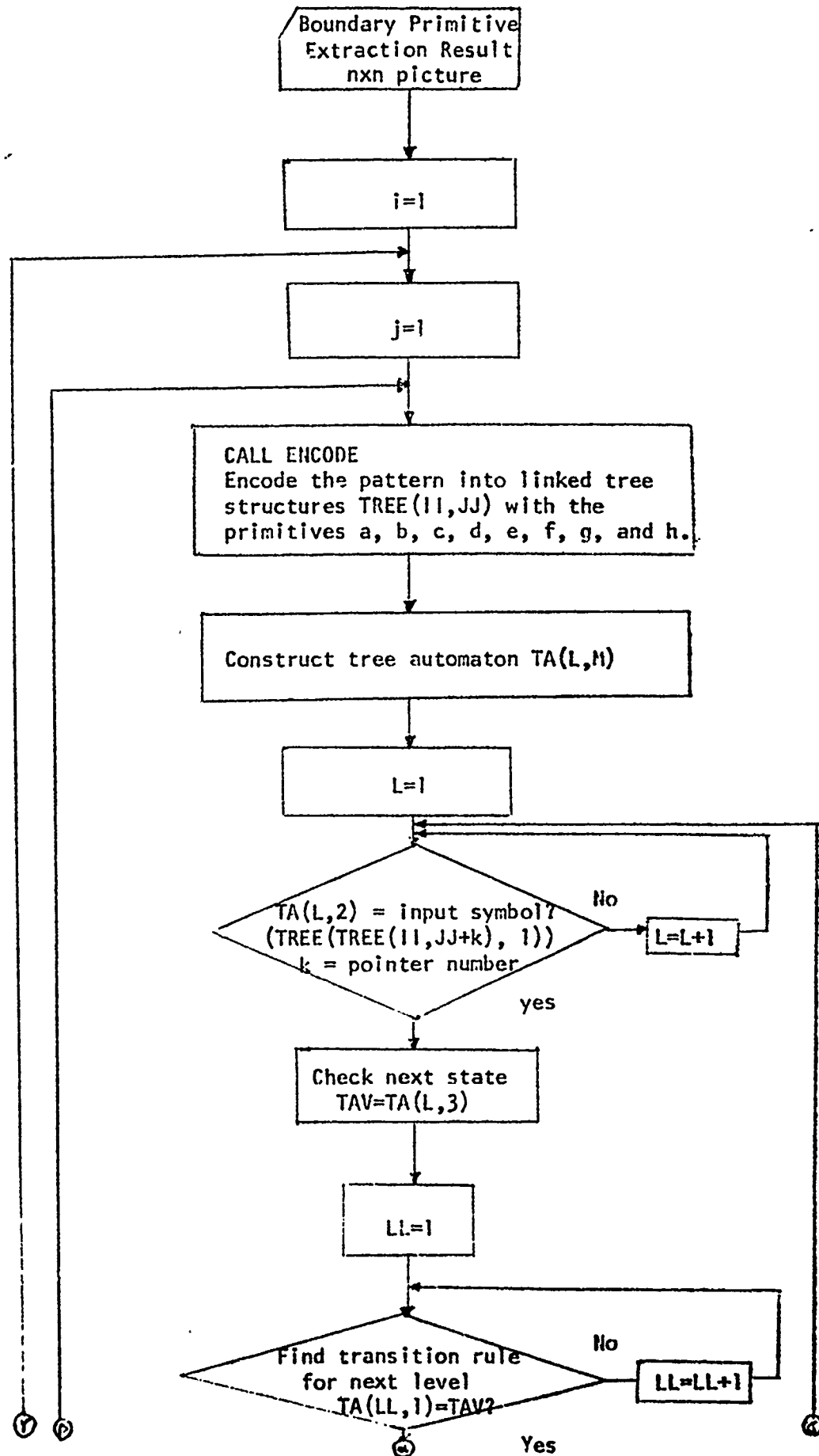
APPENDIX G

THE FLOW CHART OF TREE GRAMMAR INFERENCE PROCEDURE

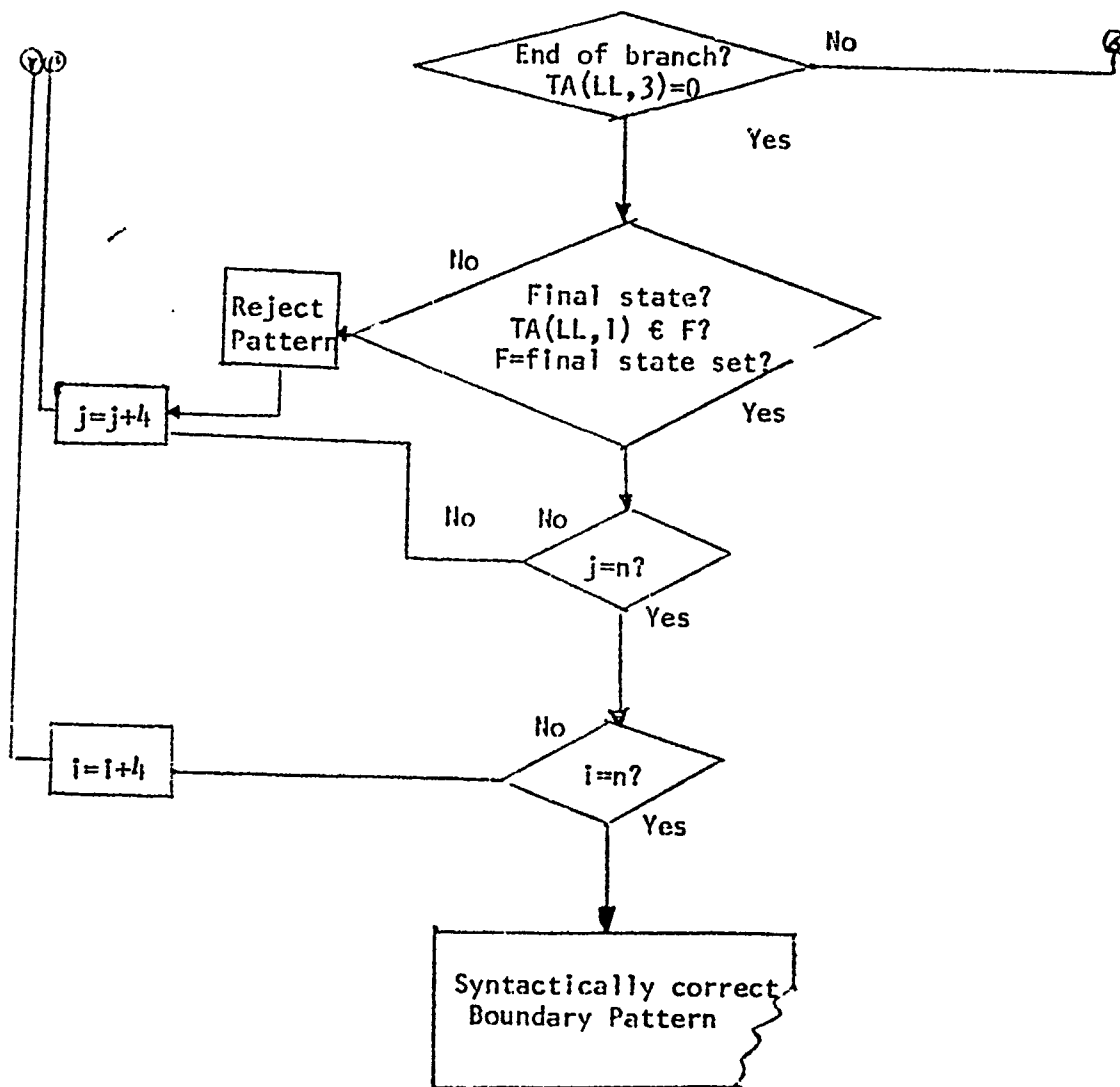


APPENDIX H

THE FLOW CHART OF TREE GRAMMAR ANALYSIS



245



APPENDIX I

THE EXAMPLE OF TREE AUTOMATON CONSTRUCTION

Example Grammar from section 4.1.1.

$$G_t = (V, r, P, S)$$

$$V = \{A_1, A_2, S, \frac{1}{2}, c, d, e, f\}$$

$$V_T = \{\frac{1}{2}, \rightarrow c, \vee d, +e, \times f\}$$

$$r(\frac{1}{2}) = \{1\} \quad r(c) = \{0, 1, 2\} \quad r(d) = \{2\}$$

$$r(e) = \{0, 1\} \quad r(f) = \{0, 1\}$$

$$P =$$

$$S \rightarrow \begin{array}{c} \frac{1}{2} \\ | \\ A_1 \end{array}$$

$$S \rightarrow \begin{array}{c} \frac{1}{2} \\ | \\ A_2 \end{array}$$

$$A_1 \rightarrow \begin{array}{c} c \\ / \quad \backslash \\ A_1 \quad A_2 \end{array}$$

$$A_2 \rightarrow \begin{array}{c} d \\ / \quad \backslash \\ A_1 \quad A_2 \end{array}$$

$$A_1 \rightarrow \begin{array}{c} c \\ | \\ A_1 \end{array}$$

$$A_2 \rightarrow e$$

$$A_2 \rightarrow \begin{array}{c} e \\ | \\ A_2 \end{array}$$

$$A_1 \rightarrow c$$

The detailed steps of tree automaton construction are as follows: the tree automaton is M_t , $M_t = (Q, f_c, f_e, f_d, f_{\frac{1}{2}}, F)$, where $Q = \{q_c, q_e, q_1, q_2, q_s\}$ and $F = \{q_c, q_e\}$.

Grammar Rule of Grammar

$$S \rightarrow \begin{array}{c} \frac{1}{2} \\ | \\ A_1 \end{array}$$

$$S \rightarrow \begin{array}{c} \frac{1}{2} \\ | \\ A_2 \end{array}$$

Transition Rule of Automaton

$$f_{\frac{1}{2}}(q_1) = q_s$$

$$f_{\frac{1}{2}}(q_2) = q_s$$

$$A_1 \rightarrow \begin{array}{c} c \\ \swarrow \searrow \\ A_1 \quad A_2 \end{array}$$

$$f_c(q_1, q_2) = q_1$$

$$A_2 \rightarrow \begin{array}{c} d \\ \swarrow \searrow \\ A_1 \quad A_2 \end{array}$$

$$f_d(q_1, q_2) = q_2$$

$$A_1 \rightarrow \begin{array}{c} c \\ | \\ A_1 \end{array}$$

$$f_c(q_1) = q_1$$

$$A_2 \rightarrow \begin{array}{c} e \\ | \\ A_2 \end{array}$$

$$f_e(q_2) = q_2$$

$$A_2 \rightarrow e$$

$$f_e(q_e) = q_2$$

$$A_1 \rightarrow c$$

$$f_c(q_e) = q_1$$

Thus the tree automaton is constructed as $M_t, M_t = (Q, f_c, f_e, f_d, f_f, F)$,

where $Q = \{q_c, q_e, q_1, q_2, q_s\}$, $F = \{q_c, q_e\}$ and f :

$$(1) \quad f_f(q_1) = q_s$$

$$(5) \quad f_c(q_1) = q_1$$

$$(2) \quad f_f(q_2) = q_s$$

$$(6) \quad f_e(q_2) = q_2$$

$$(3) \quad f_c(q_1, q_2) = q_1$$

$$(7) \quad f_e(q_e) = q_2$$

$$(4) \quad f_d(q_1, q_2) = q_2$$

$$(8) \quad f_c(q_e) = q_1$$

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR/TR-78-0694	2. GOVT ACCESSION NO. 18 AFOSR	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SYNTACTIC ALGORITHMS FOR IMAGE SEGMENTATION AND A SPECIAL COMPUTER ARCHITECTURE FOR IMAGE PROCESSING.		5. TYPE OF REPORT & PERIOD COVERED Interim rept.
6. AUTHOR(s) Janmin/Keng K. S./Fu		7. PERFORMING ORG. REPORT NUMBER TR-EE 77-39
9. PERFORMING ORGANIZATION NAME AND ADDRESS Purdue University School of Electrical Engineering West Lafayette, IN 47907		8. CONTRACT OR GRANT NUMBER(s) AFOSR-74-2661
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, DC 20332		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304 A2
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 12 274p.		12. REPORT DATE December 1977
		13. NUMBER OF PAGES 248
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		

Several efficient algorithms for image recognition and segmentation and a new computer architecture for image processing are proposed. The algorithms are "Syntactic" in that they perform structural or spatial analysis rather than statistical analysis, and a "grammar" is inferred for describing the structures of patterns in an image. Depending on the requirements of the problem, an appropriate grammatical approach is used by the syntactic algorithm.

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

292 000

1B

20. Abstract

A finite-state string grammar is applied to the image recognition of highways, rivers, bridges, and commercial/industrial areas from LANDSAT images. There are two major methods in the string grammar approach for image recognition; namely, the syntax-directed method and syntax-controlled method. For the syntax directed method, syntactic analysis is performed by a template matching which is directed by the syntactic rules. For the syntax - controlled method an automaton which is directly controlled by the syntactic rules is used for the syntactic analysis.

A tree grammar is applied to the image segmentation of terrain and tactical targets from LANDSAT and infrared images respectively. The tree grammar approach utilizes a tree automaton to extract the boundaries of the homogeneous region segments of the image. The homogeneity of the region segment is obtained through texture measurements of the image.

The computer architecture proposed is a special purpose system in that it can perform an image processing task on several picture-points of an image at the same time, and thus takes advantage of the fact that image processing tasks usually exhibit parallelism. This architecture uses a distributed computing approach. Two major features are the reconfigurable capability, and the method of computer exploitation of task parallelism. Finally, a parallel parsing scheme for tree grammar is used to demonstrate the higher efficiency of the proposed computer architecture than the conventional parsing scheme.